Universidad de La Habana
Facultad de Matemática y Computación

Extracción automática de argumentos en textos de opinión en la prensa cubana

Autor:
Luis Ernesto Ibarra Vázquez

Tutor:
MSc. Damian Valdés Santiago

Trabajo de Diploma
presentado en opción al título de
Licenciado en Ciencia de la Computación

28 de noviembre del 2022

github.com/luisoibarra/thesis

A mis amigos y familia, a todas las personas que estuvieron en mi vida dando su apoyo y ayuda para superar los obstáculos.

Agradecimientos

Este trabajo es la consolidación de varios años de estudio y esfuerzo, los cuales no habrían sido posible sin la participación y apoyo de mi familia, amigos y profesores.

A Elizabeth, René y Amanda.

A Dalmau, Miguel, Damián, Klever, Adrian, Marcos, Claudia, Laura, Yasmín, Jessy, Penélope y todos los demás con que tuve el placer de cursar estudios.

A mi tutor.

A todos ellos, ¡gracias!

Opinión del tutor

La lingüística computacional es una rama multidisciplinaria que procesa grandes cantidades de textos escritos. La tesis presentada por Luis Ernesto Ibarra Vázquez propone un algoritmo basado en modelos de aprendizaje automático para la segmentación y clasificación de enunciados argumentativos en textos de la sección "Cartas a la Dirección" del periódico *Granma*.

La tesis se ubica en el marco de un proyecto *Dinámicas sociales, políticas y económicas en el discurso público en Cuba de principio del siglo XXI: estudios de CORES-PUC*, asociado al Programa Nacional de Ciencia y Técnica "Las Ciencias Sociales y las humanidades. Desafíos ante la estrategia de desarrollo de la sociedad cubana", Código PN223LH011-011, Ministerio de Ciencia, Tecnología y Medio Ambiente (CITMA), Cuba, 2021-2023.

Por ello, Luis Ernesto tuvo que estudiar la materia referida, que no está incluida en el currículo de la carrera y trabajó mostrando creatividad, disciplina, entrega y rigor. Deseo destacar su profundización en el estudio computacional de la argumentación y la proposición de soluciones de software útiles para el análisis de enunciados argumentativos en español.

La investigación realizada por el estudiante incluyó diversos corpus en idioma inglés con diferentes anotaciones de la argumentación que se usaron para el entrenamiento de modelos de aprendizaje automático proyectivos para clasificar los enunciados argumentativos en español, a partir del conocimiento de la anotación en idioma inglés. Se compararon varios modelos y el mejor fue utilizado para la clasificación de enunciados de las Cartas a la Dirección del periódico Granma. Este trabajo es un primer paso positivo en el estudio automático de la argumentación de nuestra variante del español y llevará una posterior revisión por lingüistas. Dicho corpus anotado será importante para la realización de análisis sociolingüísticos del corpus donde la argumentación es una variable de interés para el proyecto CORESPUC.

Durante el desarrollo del trabajo, Luis Ernesto demostró habilidades para el trabajo con la bibliografía y creatividad para proponer soluciones a problemas de implementación, entre otras competencias de programación en el lenguaje Python y sus diversos *frameworks*. Así, se logró cumplir el objetivo de esta tesis.

Por tanto, considero que al estudiante Luis Ernesto Ibarra Vázquez debe otorgár-

sele la máxima calificación (5 puntos, Excelente), y estoy seguro que en el futuro Luis Ernesto se desempeñará como un excelente profesional de la Ciencia de la Computación.

MSc. Damian Valdés Santiago
21 de noviembre de 2022

Resumen

El estudio de la argumentación en la prensa cubana es un campo en donde se han reportado relativamente pocas investigaciones. En estos estudios es posible obtener información de los esquemas argumentativos utilizados en los textos y realizar acciones en base a estos. Este problema tradicionalmente es resuelto mediante una anotación manual por expertos en lingüística, trabajo que se caracteriza por tomar mucho tiempo y recursos. La Extracción de Argumentos es la rama del Procesamiento del Lenguaje Natural encargada de estudiar algoritmos y métodos para solucionar los problemas asociados a la anotación de las estructuras argumentativas. Mediante el uso de estos es posible automatizar el procedimiento de anotación de la argumentación. En este trabajo se propone la anotación de textos argumentativos mediante el uso de dos modelos de aprendizaje profundo, entrenados con conjuntos de datos traducidos y proyectados del inglés, encargados de resolver las tareas relacionadas al problema. El primer modelo propuesto consiste en uno secuencia a secuencia usado para la extracción y clasificación de las unidades de discurso argumentativas (UDA) mediante el uso de *Long Short Term Memory* (LSTM) y *Conditional Random Field* (CRF). Para la extracción y clasificación de enlaces entre las UDAs se propone un modelo de clasificación basado en redes residuales, atención y LSTM. Ambos modelos utilizan *embeddings* GloVe para la representación de las palabras. Los resultados obtenidos en la extracción de UDAs alcanzaron valores de 0,82 en la métrica F1 comparados con 0,85 obtenidos en el estado del arte. En las demás tareas, los resultados no son directamente comparables con los del estado del arte, los mejores valores F1 obtenidos fueron 0,56 en la clasificación de UDAs, 0,74 en la predicción de enlaces y 0,39 en la clasificación de enlaces. Con dichos modelos se anotaron las "Cartas a la Dirección", del periódico Granma, creándose un conjunto de datos con las estructuras argumentativas anotadas y listas para el estudio de estas por lingüistas.

Abstract

The study of argumentation in the Cuban press is a field in which relatively little research has been reported. In these studies it is possible to obtain information on the argumentative schemes used in the texts and take actions based on them. This problem is traditionally solved through manual annotation by linguistic experts, a work that takes a lot of time and resources. Argument Extraction is the branch of Natural Language Processing in charge of studying algorithms and methods to solve the problems associated with the annotation of argument structures. By using these algorithms it is possible to automate the argumentation annotation procedure. In this paper we propose the annotation of argumentative texts by using two deep learning models, trained with translated and projected English datasets, in charge of solving the tasks related to the problem. The first proposed model consists of a sequence to sequence one used for the extraction and classification of argumentative discourse units (ADUs) by using Long Short Term Memory (LSTM) and Conditional Random Field (CRF). A classification model based on residual networks, attention and LSTM is proposed for the extraction and classification of links between ADUs. Both models use GloVe for word representation. The results obtained in the extraction of ADUs reached values of 0.82 in the F1 metric compared to 0.85 obtained in the state of the art. In the other tasks, the results are not directly comparable with those of the state of the art, the best F1 values obtained were 0.56 in UDAs classification, 0.74 in link prediction and 0.39 in link classification. With these models, the "Letters to the Management" of the Granma newspaper were annotated, creating a data set with the argumentative structures annotated and ready to be studied by linguists.

Índice general

Índice de figuras

Índice de tablas

Introducción

La argumentación es una actividad verbal, social y racional destinada a convencer a un crítico razonable de la aceptabilidad de un punto de vista mediante la presentación de una constelación de proposiciones que justifican o refutan la proposición expresada en el punto de vista [Van Eemeren y Grootendorst 2004]. Esta definición concentra en ella partes esenciales de lo que es la argumentación, dicha actividad se encuentra presente en varias facetas de la cotidianedad humana, como en la escritura o lectura de documentos y en las interacciones sociales de las personas. En resumen, la argumentación está presente cada vez que se plantea un argumento y se trata de que este triunfe en un debate donde se exponen elementos que lo apoyen.

En la actualidad es necesario tener acceso a la información de forma rápida y simple. Esto no siempre es posible dado la gran cantidad de información existente y que es generada en cada momento. En caso de tener una vía de acceder a esta, se podrían realizar acciones con mayor rapidez y calidad. Con la argumentación, se podría hacer explícitas las razones de las personas al afirmar algo sobre un tema teniendo así su punto de vista individual, y con suficientes personas, colectivo.

Varias tareas en el Procesamiento de Lenguaje Natural (PLN) se han desarrollado en torno a diferentes problemas relacionados con la argumentación. Entre tales tareas se encuentran: el minado de opiniones, el cual es el estudio computacional de las opiniones, sentimientos y emociones expresadas en un texto [Liu 2010], la detección de controversias, enfocada en la identificación de temas y textos en donde puntos de vista conflictivos están presentes; y también la zonificación argumentativa que clasifica las oraciones por su rol argumentativo dentro de un artículo científico. Estas tareas solamente muestran cuáles opiniones, puntos de conflictos y roles de argumentos presenta el texto, pero lo realizan de manera separada y no muestran el porqué de estas. Para lograr extraer esta información faltante es necesario realizar un análisis de los argumentos dados, para pasar de texto no estructurado a datos argumentativos que den entendimiento de los puntos de vista y de cómo se apoyan o atacan entre sí. Este análisis es posible realizarlo mediante métodos convencionales manuales o utilizando programas especializados para la anotación, aunque la práctica ha demostrado que este proceso requiere de una gran cantidad de tiempo y de personal calificado. Con la inmensa cantidad de datos que se genera a diario este análisis es impracticable

de realizar de forma manual, por esto se estudian y crean métodos encargados de automatizar esta tarea.

La Extracción de Argumentos (EA) nace como la rama del PLN encargada del estudio de métodos para la extracción automática de las estructuras argumentativas de los textos y su posterior procesamiento [Lawrence y Reed 2020]. Esta tarea se divide en cuatro subtareas fundamentales: i) la extracción y ii) clasificación de las componentes argumentativas del texto, y iii) la extracción y iv) clasificación de las relaciones entre estas. Estas tareas han sido abordadas de diferentes maneras, desde modelos secuenciales [Palau y Moens 2009, Goudas y col. 2015] hasta *end-to-end* [Eger, Daxenberger y Gurevych 2017], desde el uso de clasificadores clásicos como Naive Bayes o SVM [Niculae, Park y Cardie 2017, Stab y Gurevych 2017] hasta el uso de aprendizaje profundo [Galassi 2021, Mayer, Cabrio y Villata 2020], y sigue siendo un área creciente de estudio.

La EA se caracteriza por la falta de conjuntos de datos y por la heterogeneidad que presentan estos a la hora de decidir cómo realizar las anotaciones, además que la gran mayoría de los estudios realizados en el campo se encuentran en un reducido conjunto de idiomas como el inglés, alemán o chino [Eger, Daxenberger, Stab y col. 2018]. En el español, pocas intervenciones se han dado en el análisis de los argumentos [Esteve, Casacuberta y Rosso 2020] y en lo correspondiente a Cuba no se encontró ninguna referencia.

Este trabajo tributa a "Dinámicas sociales, políticas y económicas en el discurso público en Cuba de principio del siglo XXI: estudios de CORESPUC". Proyecto asociado al Programa Nacional de Ciencia y Técnica "Las Ciencias Sociales y las humanidades. Desafíos ante la estrategia de desarrollo de la sociedad cubana", Código PN223LH011-011, Ministerio de Ciencia, Tecnología y Medio Ambiente (CITMA), Cuba, 2021-2023.

En esta tesis se tiene como objetivo, proponer un algoritmo para la extracción y análisis de estructuras argumentativas en textos de la prensa cubana, constituyendo el primero de su tipo en lo que respecta a la búsqueda del autor. Para lograr dicho objetivo se necesitan realizar varias operaciones. En primer lugar, es necesario obtener los textos a analizar del sitio web del periódico Granma. Luego, se necesita confeccionar algoritmos capaces de realizar las tareas de EA sobre estos textos, para estos algoritmos es necesaria la confección de conjuntos de datos en español en los cuales se puedan entrenar. Finalmente, se requiere de una interfaz visual en la que sea posible la interacción de los usuarios, en tareas como visualización y edición, con los resultados obtenidos.

Se propone la construcción de un programa capaz de realizar la extracción de los textos del periódico Granma, en específico, se enfoca en su sección de Cartas a la Dirección. Para la extracción de argumentos se presentan dos modelos, el primero se encarga de la segmentación y clasificación de las componentes argumentativas.

Este es un modelo secuencia a secuencia en el cual las palabras son vectorizadas por sus características morfológicas mediante el uso de redes neuronales convolucionales (CNN) y *Long Short Term Memory* (LSTM), por su información semántica mediante *embeddings Global Vectors* (GloVe) y por su información estructural mediante su parte de la oración, estas secuencias son procesadas por una red LSTM bidireccional y sus atributos son usados por una capa de *Conditional Random Field* (CRF) para su clasificación final en etiquetas B (inicio de segmento), I (dentro del segmento), O (afuera de segmento), E (fin de segmento) y S (segmento de un elemento) (BIOES) con información adicional para la clasificación de las unidades de discurso argumentativas (UDA). El segundo modelo está encargado de la extracción y clasificación de las relaciones entre las UDAs. En él se utilizan las representaciones GloVe de las palabras en las secuencias y su distancia argumentativa para la clasificación de una tupla en la que su primer elemento constituye el texto candidato de donde parte la relación o fuente y el segundo, el texto candidato a recibir la relación u objetivo, en el tipo de relación existente entre estos elementos. La entrada es procesada mediante una LSTM bidireccional y módulos de atención cruzada entre los elementos de la fuente y los elementos del objetivo. Para la visualización se crea un ambiente de desarrollo con la herramienta Brat [*Brat Rapid Annotation Tool* s.f.], la cual permite realizar las tareas requeridas a los datos anotados por los modelos.

En resumen, se presenta un estudio de la argumentación en español al realizar la extracción de argumentos en prensa, además se crea un conjunto de datos anotado que puede servir para un estudio más profundo sobre los esquemas argumentativos presentes en estos textos.

El presente documento está dividido en cuatro capítulos. En el Capítulo 1 se presentan las definiciones necesarias para una introducción al tema de la argumentación y la EA. El Capítulo 2 presenta un recorrido por los conceptos, métodos y arquitecturas relacionadas con el Aprendizaje Automático para el problema en cuestión, además de realizar un compendio y análisis de los enfoques utilizados para dar solución a los problemas asociados a la EA y el procedimiento de proyección de corpus, utilizado para la traducción de conjuntos de datos del inglés al español. Luego, en el Capítulo 3 se describe la propuesta de los modelos para realizar la EA. En el Capítulo 4 se mencionan los diferentes conjuntos de datos utilizados en los experimentos y los resultados obtenidos en las tareas realizadas. Finalmente, se exponen conclusiones y recomendaciones sobre el trabajo realizado.

Capítulo 1

Argumentación

En el captítulo se aborda la argumentación, definiciones y marcos teóricos existentes para el estudio de esta. Luego se introduce la Extracción de Argumentos como campo de la lingüísticas computacional encargado del estudio y procesamiento de las estructuras argumentativas de textos. También se definen y explican los componentes y las tareas asociadas al problema de realizar la extracción de argumentos.

1.1. Argumentación

La argumentación es un tema tratado desde la antigüedad, Aristóteles lo defendía como la habilidad de, dada una pregunta, considerar los elementos útiles para persuadir a alguien, algo similar a la retórica. De una perspectiva más contemporánea surgen las ideas de Perelman y col. (1969) enfocadas en un análisis de la retórica en donde se estipula que la teoría de la argumentación responde a provocar o aumentar la adhesión de las personas a las tesis presentadas, por medio de técnicas discursivas. En Toulmin (2003) se considera como argumento todo aquello que ofrece, o todo lo que es utilizado, para justificar o refutar una proposición. En este último, se toma una perspectiva más racional y deductiva de la argumentación, dando como resultado lo que se conoce como el Método de Toulmin.

1.1.1. Método de Toulmin

Este método divide los argumentos en seis partes: afirmación (*claim*), fundamento (*grounds*), justificación (*warrant*), calificador (*qualifier*), refutación (*rebuttal*) y respaldo (*backing*). Mediante las afirmaciones se conoce el argumento principal que el autor quiere probar a la audiencia, estas son respaldadas con fundamentos, siendo estos las evidencias y hechos en que se apoya el autor. Las justificaciones pueden estar explícitas o implícitas y son suposiciones que vinculan los fundamentos con las

afirmaciones, estas a su vez pueden ser respaldadas por conocimiento. El esquema introduce la posibilidad de otra situación válida a la establecida en las afirmaciones mediante la refutación. Los calificadores son usados para dar más información de la calidad o seguridad de las afirmaciones dadas. Un ejemplo[1] de este esquema es:

> [*Se escucharon ladridos y aullidos en la distancia*]$_{\text{fundamento}}$, [*probablemente*]$_{\text{calificador}}$ [*haya perros en las cercanías*]$_{\text{afirmación}}$.

En este ejemplo, además de las partes explícitas, se encuentran partes implícitas como la justificación (*los perros son animales que ladran y aúllan*), el respaldo (*se sabe que existen perros en la zona*) y la refutación (*puede ser que hayan lobos o coyotes cerca*).

Este método crea una definición compacta que ayuda a los investigadores a enfocar su búsqueda en las diferentes categorías definidas. Además, engloba de manera comprensible un tema tan complejo como la argumentación al tomar en cuenta gran parte de los elementos presentes en el razonamiento realizado para llegar a conclusiones, incorporando incluso elementos probabilísticos en el proceso.

1.1.2. Rasgos lingüísticos

Los rasgos lingüísticos son aquellas características que se encuentran presentes en los textos que hacen que estos se clasifiquen argumentativos [Venegas 2005]. Con la identificación de estos se hace la tarea de extracción más sencilla y con un marco teórico que respalde las decisiones tomadas. Ejemplos de rasgos presentes en textos argumentativos:

1. Marcas de orden que introducen párrafos: *primero*, *segundo*, *por un lado*, *por otra parte*, *finalmente*.

2. Comillas y citas: citar palabras que refuercen la intervención recurriendo a autoridades o personajes.

3. Nexos que expresan causa o consecuencias: *ya que*, *porque*, *pues*, *con motivo de*, *gracias a*, *considerando que*, *por lo tanto*, *de manera que*.

Estos rasgos además de dar indicación de la existencia de argumentos dan pie para conocer las relaciones entre estos y los tipos de argumentos. Por ejemplo, *por lo tanto*, implica que lo que viene a continuación es una conclusión apoyada en lo dicho anteriormente en el texto. Algo parecido sucede con *ya que*, en este caso implica que lo siguiente es un argumento que se encuentra relacionado con lo mencionado antes.

En Venegas (2005) se determinan 16 categorías y 51 rasgos lingüísticos, dando una idea de la gran variedad de marcadores presentes en la argumentación.

[1] Extraído de *Historical Perspectives on Argumentation - Toulmin Argument* (2022).

1.2. Extracción de Argumentos

El PLN es un subcampo de la Inteligencia Artificial que tiene como objetivo la comprensión del lenguaje humano por las computadoras. Mediante el uso de sus algoritmos es posible el procesamiento masivo de texto para la extracción de información relevante de este. Entre las tareas pertenecientes a dicho campo se encuentran la Traducción Automática, la Generación de Lenguaje Natural y la Extracción de Argumentos (EA). La EA constituye la identificación y extracción automática de las estructuras de inferencia y razonamiento expresadas como argumentos presentes en el lenguaje natural [Lawrence y Reed 2020]. En la actualidad, tareas del PLN como el análisis de sentimientos permiten extraer cuáles son las opiniones o sentimientos presentes, sin embargo, este análisis presenta una falta de información, ya que no justifica el porqué de las opiniones. La EA permite dar respuesta a este problema presentando los argumentos y cómo sus relaciones justifican las posiciones del hablante. Dicho problema está constituido por diferentes estructuras y se compone de distintas tareas necesarias para su solución.

1.2.1. Estructuras Argumentativas

Las estructuras argumentativas son las partes de la argumentación de los textos y sus relaciones. Estas se componen de dos elementos principales: las Unidades de Discurso Argumentativas (UDA) y los enlaces existentes entre estas. Las UDAs corresponden a la unidad mínima de argumentación, definida como un segmento de texto que juega un solo rol para el argumento analizado, y es delimitado por segmentos vecinos que tienen roles diferentes o ningún rol [Stede y Schneider 2018]. Las UDAs se relacionan entre sí conformando el proceso de inferencia y razonamiento del argumento. Tanto los enlaces como las UDAs son clasificados en dependencia de su rol en la argumentación, estas clasificaciones en las UDAs parten de los conceptos de afirmación, declaración controversial y parte central del argumento, y premisa, razones que la justifican o refutan, y en las relaciones de ataque y apoyo.

1.2.2. Tareas de extracción de argumentos

Dada la definición de estructuras argumentativas y que el objetivo de la EA es extraerlas, se conciben las siguientes tareas principales:

Extracción de UDAs

Consiste en la separación de los segmentos de texto que formarán parte de la estructura. En este proceso el texto es segmentado y se obtiene un conjunto de UDAs.

En el siguiente ejemplo[2] se representa la extracción de UDAs marcadas en *cursiva* de un texto dado:

> En primer lugar, [*el correo electrónico puede contar como uno de los resultados más beneficiosos de la tecnología moderna*]. [*Años atrás, las personas pagaban gran cantidad de dinero para enviar sus cartas y sus pagos estaban sujetos al peso de sus cartas o paquetes y muchos accidentes podrían causar problemas que causarían que el correo no fuera enviado*].

Clasificación de UDAs

La clasificación de UDAs consiste en asignarle la categoría que toma la UDA en la argumentación. En general, las clasificaciones parten de dos clases bases, las afirmaciones y premisas, aunque estas pueden ser tantas como sea necesario por el problema específico a tratar. Siguiendo con el ejemplo, se observa la clasificación en afirmación y premisa asignada a los segmentos extraídos en el paso anterior:

> En primer lugar, [*el correo electrónico puede contar como uno de los resultados más beneficiosos de la tecnología moderna*]$_{\text{Afirmación}}$. [*Años atrás, las personas pagaban gran cantidad de dinero para enviar sus cartas y sus pagos estaban sujetos al peso de sus cartas o paquetes y muchos accidentes podrían causar problemas que causarían que el correo no fuera enviado*]$_{\text{Premisa}}$.

Extracción de relaciones entre las UDAs

La extracción de relaciones constituye el paso donde se determina si están relacionadas las UDAs o no. La disposición de estas relaciones forma el proceso de razonamiento en que se basa el autor para validar su posición. En el ejemplo se representa la existencia de relación mediante su distancia argumentativa con la UDA con la que se relaciona. La distancia argumentativa es la cantidad de UDAs del texto que separan la UDA fuente del objetivo [Galassi 2021], en caso de ser negativa (positiva) el objetivo se encuentra antes (después) que la fuente.

> En primer lugar, [*el correo electrónico puede contar como uno de los resultados más beneficiosos de la tecnología moderna*]$_{\text{Afirmación}}$. [*Años atrás, las personas pagaban gran cantidad de dinero para enviar sus cartas y sus pagos estaban sujetos al peso de sus cartas o paquetes y muchos accidentes podrían causar problemas que causarían que el correo no fuera enviado*]$_{\text{Premisa},-1}$.

Clasificación de relaciones entre las UDAs

La clasificación de las relaciones consiste en clasificar las relaciones extraídas en las categorías pertinentes. Los tipos de relaciones, nacen de dos clases bases por lo

[2]Traducido del corpus de Stab y Gurevych (2017) .

general, las relaciones de apoyo y de ataque. Las de apoyo son aquellas en las que la UDA fuente afirme la UDA objetivo, las de ataque son las que la UDA fuente apoya la negación de la UDA objetivo.

> En primer lugar, [*el correo electrónico puede contar como uno de los resultados más beneficiosos de la tecnología moderna*]$_{\text{Afirmación}}$. [*Años atrás, las personas pagaban gran cantidad de dinero para enviar sus cartas y sus pagos estaban sujetos al peso de sus cartas o paquetes y muchos accidentes podrían causar problemas que causarían que el correo no fuera enviado*]$_{\text{Premisa,}-1\text{,apoyo}}$.

Partiendo de esto, se puede observar que las estructuras argumentativas de un texto constituyen un grafo dirigido en donde sus nodos representan las UDAs y están anotados con su tipo, y sus vértices representan las relaciones entre las UDAs. Dichos vértices se anotan con el tipo de relación existente entra ambas (Figura 1.1).

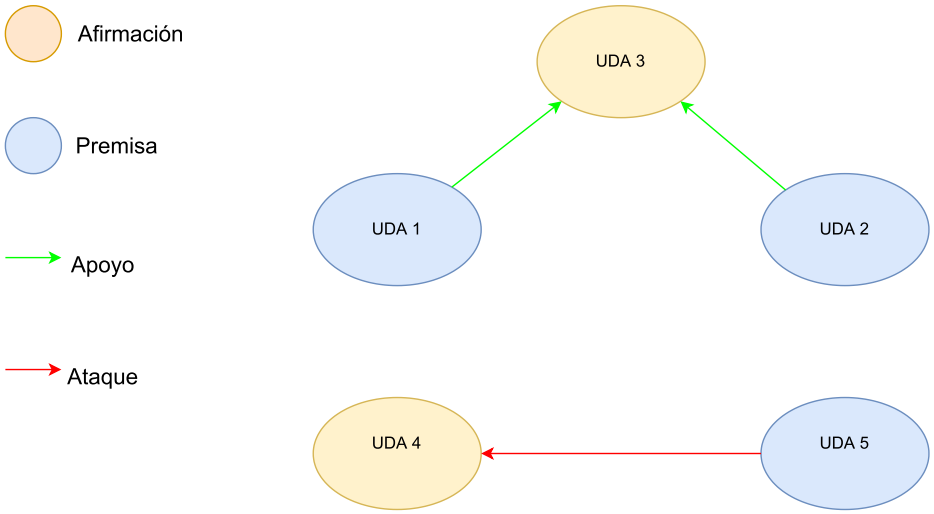

Figura 1.1: Estructuras Argumentativas.

Capítulo 2

Marco teórico

En este capítulo se introducen conceptos de Aprendizaje Automático (AA). Temas como la representación de datos, arquitecturas y evaluación de modelos son tratados en sus secciones. Luego, se hace un recorrido por las diferentes investigaciones recientes realizadas en la EA, viendo los enfoques desarrollados y la evolución de estos a partir del desarrollo e introducción de nuevos métodos en el campo. Por último, se introduce la proyección de corpus, técnica necesaria para la creación de conjuntos de datos para el entrenamiento de modelos de AA.

2.1. Aprendizaje Automático

Existen varios problemas para los cuales es difícil escribir un programa de forma tradicional que pueda resolverlo, por ejemplo, decir si en una imagen existe un gato o un perro, o transcribir una grabación. En el caso de que se escribiera este programa probablemente sería frágil y poco escalable. El AA constituye el estudio de técnicas que pueden aprender de la experiencia [Zhang y col. 2021] y mediante su uso se puede automatizar el proceso de encontrar soluciones a dichos problemas, haciendo que sus resultados sean robustos y escalables.

En AA existen tres tipos de aprendizaje:

- Aprendizaje no supervisado: trabaja con datos no anotados y los algoritmos tratan de extraer información de estos.
- Aprendizaje reforzado: trabaja en la creación de agentes que interactúen con un ambiente obteniendo información de qué tan buenas son las acciones realizadas.
- Aprendizaje supervisado: trabaja con datos anotados y los algoritmos tratan de minimizar el error cometido en las predicciones.

A este último principalmente se refiere esta sección.

Los componentes básicos de un problema de aprendizaje supervisado se pueden resumir en los datos de los que aprender, el modelo para transformar los datos, una función de pérdida que cuantifica qué tan malo o bueno es el modelo y un algoritmo que ajuste los parámetros del modelo para minimizar la pérdida.

Matemáticamente, un modelo constituye una función $\hat{y} = f_\theta(x)$ cuyo argumento son las representaciones de las entradas originales x y devuelve una salida $\hat{y}$. Esta función f se encuentra parametrizada por el vector θ el cual constituye los parámetros en los que el modelo se basa para realizar sus cálculos. La función de pérdida se puede definir como $e(y, \hat{y})$, donde y es el resultado esperado para x. El objetivo final del algoritmo de ajuste es encontrar θ tal que:

$$\arg\min_{\theta} e(y, f_\theta(x)). \tag{2.1}$$

Los parámetros θ eventualmente se van ajustando con algún algoritmo de optimización para que la función de error alcance un valor óptimo, aunque un óptimo global en la práctica es difícil de conseguir debido a la complejidad del espacio de búsqueda.

Una manera de observar la complejidad de estos modelos sería la cantidad de capas de procesamiento que lo integran. Los primeros modelos empezaron utilizando una sola capa[1] para realizar las tareas, más tarde estos modelos fueron ganando en complejidad al superponerle otras, dando paso al aprendizaje profundo (*Deep Learning*). La superposición se refiere a la composición de funciones f_i y f_{i+1} de tal forma que la imagen de f_i sea compatible con el dominio de f_{i+1}, entonces se obtiene $f_{i+2}(x) = f_{i+1}(f_i(x))$, al aplicar este esquema se pueden agregar múltiples capas y profundizar f tanto como se desee.

2.1.1. Representación de datos

La representación de los datos constituye una parte importante al modelar un problema. Esta es la encargada de presentar datos abstractos, como imágenes, párrafos o sonidos, en formas tratables por los algoritmos. Generalmente, se buscan configuraciones que recojan la mayor cantidad de información de la entrada relevante al problema.

En el PLN los datos suele estar representados con distintos niveles de granularidad. De menor granularidad a mayor se pueden ir mencionando los documentos, párrafos, oraciones, palabras y caracteres. Estos elementos es posible representarlos mediante vectores que codifiquen propiedades objetivas del problema a tratar como características morfológicas o semánticas de estos. Por ejemplo, los textos tienen una

[1] A estos modelos se les conoce como *shallow* o poco profundos.

popular manera de representarse mediante vectores de TF-IDF [Manning 2008], una desventaja de esta representación es que no toma en cuenta el orden de las palabras, por lo que la representación de *no me gusta,* y *no, me gusta* serían iguales aunque semánticamente sean opuestas. Para agregarle la información de orden a la representación se introducen los llamados $n-$gramas, los cuales toman en cuenta una ventana de tamaño n para conformar conjuntamente una representación. Este enfoque conlleva a la limitante de que solo un contexto finito está disponible para hacer las inferencias, para esto finalmente se crean representaciones individuales para cada palabra, las cuales codifican información y la secuencia completa es introducida al modelo para realizar la inferencia. La manera más simple de representar una palabra es mediante la representación *one-hot*, en esta, a cada palabra se le asigna un índice y el vector resultante posee la dimensión del vocabulario y es rellenado con ceros en todos sus elementos excepto en el índice de la palabra, donde se le asigna 1. Esta representación asume que las palabras son independientes entre sí y computacionalmente ocupan un espacio considerable. Existen otras representaciones que brindan información al algoritmo sobre la morfología y la semántica de la palabra que representa.

Las características morfológicas son aquellas que describen cómo está formado el elemento a analizar. Estas pueden ser extraídas con relativa facilidad, entre tales características se encuentran: tamaño, cantidad, posición de palabras o párrafos, presencia de sufijos, prefijos, acentos u otros marcadores en el texto. Las características semánticas presentan una mayor dificultad a la hora de ser extraídas. Para esto se utilizan diferentes modelos que codifican esta información en vectores continuos conocidos como *embeddings*, entre los modelos existentes se encuentran word2vec [Mikolov y col. 2013], *Global Vectors* (GloVe) [Pennington, Socher y Manning 2014], *Bidirectional Encoder Representations from Transformers* (BERT) [Devlin y col. 2018], entre otros.

2.1.2. Modelación de problemas

Los problemas en la vida real se presentan en un principio como una descripción a un nivel de abstracción más alto que el requerido por los algoritmos existentes, se hablan de entidades abstractas como imágenes, sonidos, textos o usuarios, además de la información que se desea extraer o las operaciones que se desean aplicar sobre estas entidades. A lo largo del desarrollo del AA han aparecido diferentes tipos de problemas que aparecen frecuentemente en la práctica y sus respectivas maneras de solucionarlos. Uno de estos tipos de problemas es el de secuencia a secuencia (*seq2seq*), cuyo objetivo es la conversión de una secuencia en un dominio de entrada a otra secuencia en un dominio de salida. Con este tipo de problemas se modelan tareas como traducción automática, en donde el dominio de entrada es texto en un lenguaje de partida y el dominio de salida es texto en un lenguaje de llegada. Otro problema que se resuelve

con *seq2seq* es la segmentación de texto, cuya entrada es una lista de *tokens* y la salida son etiquetas que indican cuándo empieza y termina un segmento de texto. Las etiquetas utilizadas para la segmentación, generalmente, son:

- B (*begin*): representa el inicio del segmento.
- I (*inside*): representa la continuación del segmento previamente iniciado.
- O (*outside*): representa la no pertenencia a un segmento.

Este esquema presenta una versión más elaborada en la que se agregan dos clases que representan el final de un segmento E (*end*) y un segmento de un solo elemento S (*single*). La tarea anterior es utilizada como base para otra un poco más compleja, en la que, además de segmentar el texto, se desea clasificar sus segmentos. Para este problema se añaden meta etiquetas correspondientes a los tipos a identificar C obteniendo un conjunto final $R = \{B, I, E, S\} \times C \cup \{O\}$. En tareas como la extracción de entidades nombradas este método es empleado, donde el conjunto C contiene las posibles categorías de entidades a identificar.

2.1.3. Arquitecturas

En aprendizaje profundo existen una gran cantidad de arquitecturas que se pueden utilizar para formar el modelo final. Estas deben de ser seleccionadas en dependencia de los datos y el problema a tratar, ya que sus diseños emulan diferentes operaciones sobre datos que pueden ser más beneficiosas en situaciones específicas.

Capas densas

El perceptrón consiste es una transformación lineal del vector de datos $\mathbf{x}$ con un sesgo b y luego aplicar una transformación no lineal g, conocida como función de activación, para la obtención del resultado final:

$$f(\mathbf{x}) = g(\mathbf{w}\mathbf{x} + b). \tag{2.2}$$

El perceptrón constituye la unidad básica de las capas densas, ya que estas consisten en la aplicación de este modelo varias veces sobre la misma entrada $\mathbf{x}$ produciendo vectores de la dimensión k deseada como salida final. Los parámetros se codifican en la matriz $\mathbf{W}$ y sus sesgo en el vector $\mathbf{b}$:

$$f(\mathbf{x}) = g(\mathbf{W}\mathbf{x} + \mathbf{b}). \tag{2.3}$$

Para las funciones de activación existen varias elecciones. Una de estas es la función sigmoidal. Esta devuelve un valor entre 0 y 1, es usada en tareas de regresión logística. Se puede interpretar como el nivel de activación de la neurona:

$$sigm(x) = \frac{1}{1 - e^{-x}}. \tag{2.4}$$

Rectified Linear Unit (ReLU) se define como la parte positiva del argumento. Una ventaja que trae esta función de activación es su rápido cálculo de su derivada y que previene en parte de los problemas de desaparición de gradiente de la sigmoidal:

$$relu(x) = \text{máx}(0, x). \tag{2.5}$$

La función de activación *softmax* es diferente a las anteriores en el sentido que necesita el vector salida de la capa anterior para ser computada. Esta función convierte las q salidas en una distribución de probabilidad, algo necesario en tareas de clasificación:

$$softmax_k(\mathbf{x}) = \frac{e^{x_k}}{\sum_{i=1}^{q} e^{x_i}}. \tag{2.6}$$

Redes Neuronales Convolucionales

Las redes neuronales convolucionales (CNN, en inglés) son un tipo de redes usadas principalmente para tratar datos donde su estructura espacial es relevante, por ejemplo, en datos bidimensionales como imágenes y en unidimensionales como sonido y texto. Estas redes aplican de una función de kernel sobre los datos, $f * g$ donde f son los datos, g es el kernel o filtro y $*$ es el operador de convolución [Zhang y col. 2021]. La función g se puede aprender en el proceso o también puede ser una función de agrupación predefinida. Un ejemplo bidimensional de una función de kernel se muestra en la Figura 2.1.

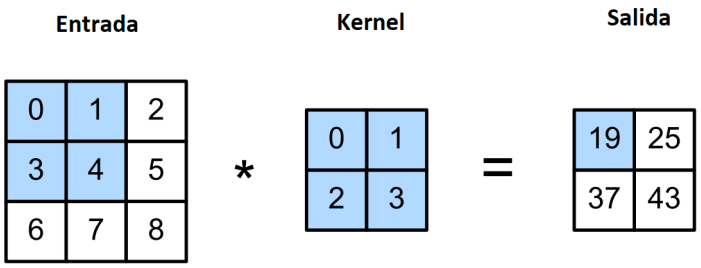

Figura 2.1: Convolución con kernel (Tomado de Zhang y col. (2021) pág. 241).

En este se observa cómo una nueva representación es computada al operar el kernel por la matriz de datos. Este corrimiento se puede realizar de diferentes formas, por

ejemplo, se puede mover de dos en dos en vez de uno en uno, este parámetro se le conoce como tamaño de paso (*stride*). Es posible además preservar las dimensiones iniciales de los datos al aplicarle un aumento de los datos en los bordes de tal forma que el resultado sea de la misma dimensión, este aumento se realiza, generalmente, rellenando convenientemente los espacios con ceros, a esto se le llama *padding* (Figura 2.2).

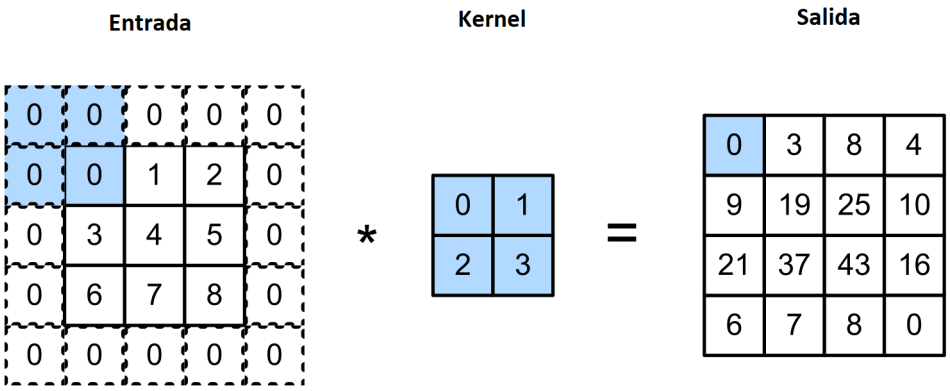

Figura 2.2: Convolución con kernel con *padding* (Tomado de Zhang y col. (2021) pág. 241).

Existen varias funciones de agrupación usadas. Entre estas se encuentran las de agrupación máxima y de agrupación media. Como sus nombres indican, la de agrupación máxima devuelve el valor máximo de los encontrados en la ventana del kernel, la de media calcula el promedio de estos valores. Estas capas tienen la capacidad de obtener información resumida sobre los datos.

Redes Residuales

Al crear modelos de aprendizaje profundo se tienen un conjunto de parámetros θ. Las posibles combinaciones de estos forman un espacio de funciones F al cual pertenecen todas las posibles instancias del modelo. Agregar nuevas capas aumenta la complejidad de este, pero no hay garantía de que el viejo espacio de funciones F sea subconjunto del nuevo espacio F', lo que implica que el nuevo modelo no es necesariamente estrictamente superior al antiguo. Este problema es la razón para la aparición de las Redes Residuales. Una red residual está formada por uno o varios bloques residuales, en los que a la salida de cada bloque residual le es sumada la entrada de este mediante una conexión residual. El objetivo de realizar tal operación es que es posible hacer la contribución del bloque 0 obteniendo así un modelo equivalente

a uno sin el bloque, garantizando la condición de subconjunto $F \subset F'$, además, dicho bloque no pierde poder expresivo, dado que en caso de que su aporte al resultado final sea considerable, se tendría que aprender solamente la función $f(x) - x$ donde x es la entrada del bloque y f es la función aprendida por el bloque sin la conexión residual, para mitigar el efecto de la conexión residual como se muestra en la Figura 2.3.

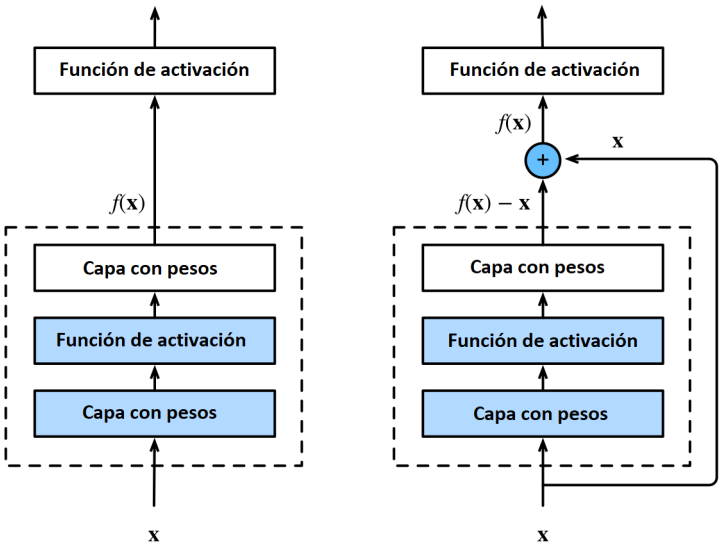

Figura 2.3: Bloque residual (Tomado de Zhang y col. (2021) pág. 289).

Redes Neuronales Recurrentes

Las redes neuronales recurrentes (RNN, en inglés) son un tipo especial de arquitectura especializada en el trabajo con datos secuenciales. Este tipo de arquitectura presenta variables en las que se almacenan información pasada, que es usada para el computo de la salida. El problema se puede modelar probabilísticamente mediante la estimación de $P(x_t|x_{t-1}, \ldots, x_1)$, donde existen dos variantes principales. En una variante se fija un tamaño de ventana α en el tiempo, dando como resultado $P(x_t|x_{t-1}, \ldots, x_{t-\alpha})$, a este tipo de modelos se les conoce como autorregresivos. Otra estrategia consiste en guardar un contexto de observaciones pasadas h_t y con este realizar la estimación $P(x_t|h_t)$, el contexto se actualiza en cada paso mediante una función $h_t = g(h_{t-1}, x_{t-1})$, a estos se les nombra modelos autorregresivos latentes, debido a la existencia de variables ocultas h_t.

En la práctica estos modelos presentan problemas de gradientes, ya que estas pueden volverse extremadamente grandes o desaparecer. Para esto se han creado ar-

quitecturas que disminuyen estos problemas. Una de estas arquitecturas es las de memorias de corto largo plazo (LSTM, en inglés) [Hochreiter y Schmidhuber 1997]. Este modelo guarda un contexto del procesamiento y está constituido por varias compuertas que regulan las actualizaciones de los estados internos. LSTM (Figura 2.4) posee dos variables de estado, la memoria C y el estado oculto H. Entre sus compuertas se encuentran la compuerta de olvido, esta regula cuánto de la memoria permanece en el próximo paso, la compuerta de entrada ajusta la cantidad de información nueva que entrará, la compuerta de salida maneja el cálculo del próximo estado oculto.

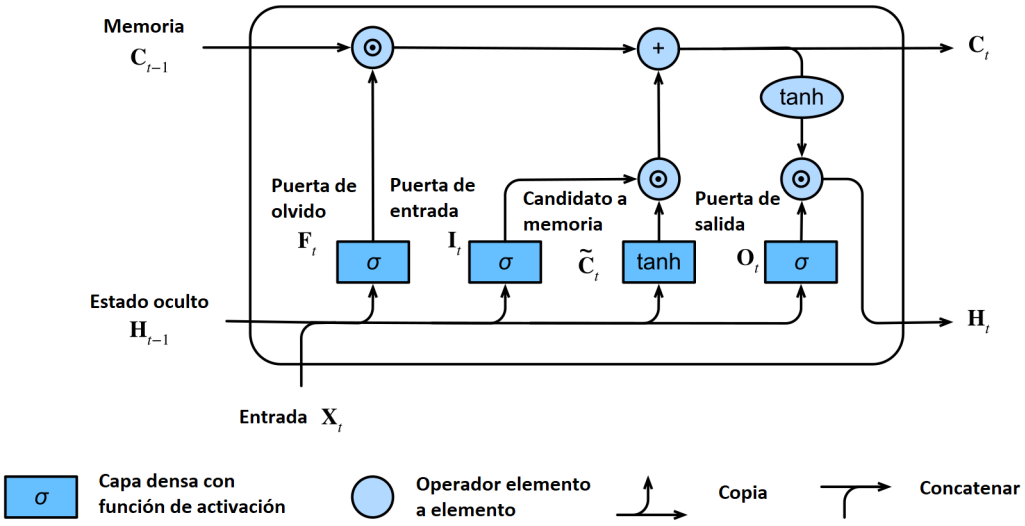

Figura 2.4: LSTM (Tomado de Zhang y col. (2021) pág. 357).

El método de aprendizaje de las RNN solamente observa los elementos anteriores de la secuencia, aunque existen tareas en las que, observando los elementos posteriores, se brinda más contexto e información a la tarea, sin que interfiera en el proceso de inferencia. El modelo bidireccional presenta una alternativa para tratar con este tipo de problemas, este modelo consiste en, además de hacer el recorrido de inicio a final de la secuencia, realizar otro recorrido en orden inverso (Figura 2.5), estos recorridos van generando dos estados ocultos $\overrightarrow{H}_i$ y $\overleftarrow{H}_i$ que luego son mezclados para obtener el contexto final H_i, los tipos de mezclas comunes son la concatenación de los estados o la multiplicación elemento a elemento de estos.

Atención

La atención es una técnica en la cual se hace una selección ponderada de atributos en un contexto específico. Este mecanismo presenta dos partes, una consulta q y una

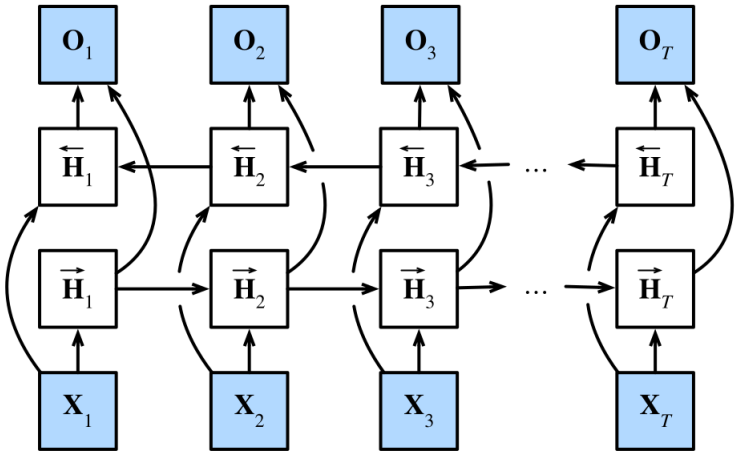

Figura 2.5: Red neuronal bidireccional (Tomado de Zhang y col. (2021) pág. 367).

colección de pares llave-valor, (k_i, v_i). La consulta representa el contexto en donde se quiere aplicar la atención y las llaves k_i son elementos que relacionan la consulta a los valores v_i. El proceso de calcular el resultado consiste en, primero, calcular el vector compatibilidad e entre las llaves y la consulta mediante la función f, este vector es luego modificado por una función g que distribuye los valores obteniendo el vector atención a. Finalmente este vector es utilizado para calcular el resultado final al aplicarle la función $o = z(a, V)$:

$$e = f(q, K), \tag{2.7}$$

$$a = g(e), \tag{2.8}$$

$$o = z(a, V). \tag{2.9}$$

En dependencia de cómo se seleccionen las funciones f, g y z se pueden obtener distintos tipos de atención. Una configuración simple consiste en definir f como el producto punto de la consulta con la llave, g como *softmax* y z la suma ponderada de v_i con los valores de atención.

Campo Aleatorio Condicional

El campo aleatorio condicional (CRF, en inglés) es un tipo de modelo gráfico probabilístico que trabaja eficientemente con secuencias, modelando conjuntamente

la probabilidad de las etiquetas de sus elementos dadas sus observaciones [Lafferty, McCallum y Pereira 2001]. En trabajos de secuencias, la forma más simple que toma el grafo consiste en una cadena de las variables representando las etiquetas de las secuencias Y, conectadas de la forma (Y_i, Y_{i+1}) y las variables observadas X, conectadas a las variables Y [Wallach 2004].

El objetivo de CRF es calcular la secuencia Y^* tal que:

$$Y^* = \arg\max_{Y} P(Y|X). \tag{2.10}$$

En esta expresión se observa que devuelve la secuencia más probable, dadas las variables observadas o atributos X, por lo que esta capa es usada al final del proceso para problemas de clasificación de secuencias.

2.1.4. Evaluación del modelo y métricas

Los modelos de AA necesitan maneras de expresar qué tan buenos son en las tareas encomendadas. Para esto se crean funciones que evalúan los resultados obtenidos por dichos modelos, estas funciones se les da el nombre de métricas. Existen diferentes tipos de métricas para tratar con diferentes tipos de problemas. En aprendizaje supervisado una métrica se define como una función $m_s(Y, \hat{Y})$, donde Y son las predicciones verdaderas y $\hat{Y}$ son las predicciones hechas por el modelo. En algoritmos de aprendizaje no supervisado como K-Means y K-NN son usadas funciones $m_{ns}(\hat{Y})$ donde $\hat{Y}$ son las predicciones finales. En comparación con su versión supervisada estas funciones no tiene acceso a las predicciones verdaderas del problema.

Clasificación

En problemas de clasificación (problemas en donde las etiquetas a predecir son discretas) son empleadas medidas que toman en cuenta la naturaleza discreta de su conjunto imagen. Medidas como precisión, recobrado, *accuracy* y F1 son utilizadas en la evaluación de los resultados, mientras que como función de error se usa entropía cruzada (*cross entropy* en inglés) [Grandini, Bagli y Visani 2020].

La matriz de confusión es una vía de representar los resultados de dos clasificadores. Esta matriz en M_{ij} indica la cantidad de elementos que clasificó como clase i el primer clasificador y como clase j el segundo clasificador. En su uso práctico, un clasificador son las etiquetas verdaderas mientras que el otro es el clasificador que se está evaluando. En problemas de la clasificación binaria, donde se busca saber si existe pertenencia o no de un elemento a una clase, se pueden observar los siguientes casos:

- Verdaderos Positivos (VP): elementos clasificados correctamente que pertenecen a la clase.
- Verdaderos Negativos (VN): elementos clasificados correctamente que no pertenecen a la clase.
- Falsos Positivos (FP): elementos que no pertenecen a la clase clasificados incorrectamente en que pertenecen.
- Falsos Negativos (FN): elementos que pertenecen a la clase clasificados incorrectamente en que no pertenecen.

Tabla 2.1: Matriz de confusión binaria.

Clases	Positivo	Negativo
Positivo	VP	FN
Negativo	FP	VN

La precisión es la medida que indica la probabilidad de que la clasificación de una clase sea correcta. Esto se puede observar como la proporción de los elementos correctamente clasificados sobre el total de elementos clasificados:

$$prec_i = \frac{VP}{VP+FP}. \tag{2.11}$$

En problemas de clasificación múltiple surge la versión macro de esta medida calculada como la media de todas las precisiones de las clases existentes:

$$prec_{macro} = \sum_{i=1}^{K} \frac{prec_i}{K}. \tag{2.12}$$

El recobrado es la medida que indica la probabilidad de que se clasifique correctamente un elemento de la clase del total existente. Esto se puede observar como la proporción de los elementos correctamente clasificados sobre el total de elementos que pertenecen a la clase:

$$rec_i = \frac{VP}{VP+FN}. \tag{2.13}$$

En problemas de clasificación múltiple surge la versión macro de esta medida calculada como la media de todos los recobrados de las clases existentes:

$$rec_{macro} = \sum_{i=1}^{K} \frac{rec_i}{K}. \tag{2.14}$$

La medida F1 es la media armónica de la precisión y el recobrado. En esta la contribución de la precisión y el recobrado al resultado final es el mismo, aunque es posible buscar variaciones de acuerdo a al problema a tratar:

$$F1_i = 2\frac{prec_i \cdot rec_i}{prec_i + rec_i}. \tag{2.15}$$

En problemas de clasificación múltiple surge la versión macro de esta medida calculada la propia medida F1, pero utilizando la precisión y recobrado macro del problema:

$$F1_{macro} = 2\frac{prec_{macro} \cdot rec_{macro}}{prec_{macro} + rec_{macro}}. \tag{2.16}$$

La métrica α%F1 [Persing y Ng 2016] es una métrica basada en la idea de F1 orientada para el trabajo con secuencias donde α denota el porcentaje de secuencia inferida que debe coincidir con la secuencia anotada para ser considerado una coincidencia. Esta versión permite establecer el rango de flexibilidad si $\alpha = 100$ (100%F1), significa que deben coincidir completamente, mientras si $\alpha = 50$ (50%F1) significa que, si coinciden en una proporción mayor o igual a la mitad, se considera como un verdadero positivo. La definición de los valores de verdaderos positivos, negativos y falsos negativos es:

$$VP = |\{j|\exists i|gl(j) = pl(i) \wedge i = j\}|, \tag{2.17}$$

$$FP = |\{i|pl(i) \neq n \wedge \nexists j|gl(j) = pl(i) \wedge i = j\}|, \tag{2.18}$$

$$FN = |\{j| \nexists i|gl(j) = pl(i) \wedge i = j\}|, \tag{2.19}$$

donde i y j son las UDAs extraídas, $gl(j)$ es la etiqueta correcta para j, $pl(i)$ es la etiqueta inferida para i, n es la clase no argumentativa, $i = j$ significa que i es una coincidencia para j.

Las métricas anteriores están acotadas por los valores 0 y 1, donde 1 representa la mejor evaluación y 0 la peor.

La entropía cruzada se encarga de evaluar qué tan diferentes son dos funciones de distribución p y q, su resultado es un número no negativo que, a medida que sean más pequeños los valores, indican mayor similitud. En su versión discreta se formula así:

$$H(p,q) = -\sum_{x \in D} p(x) \log q(x). \tag{2.20}$$

Cadenas

Para textos o cadenas existen diferentes métricas que constituyen formas de saber la similitud entre dos elementos. Una de esas métricas es la similitud de Jaccard, esa se puede ver como la proporción de elementos comunes que presentan dos conjuntos, los elementos sería palabras:

$$jac(X,Y) = \frac{|X \cap Y|}{|X \cup Y|}. \tag{2.21}$$

Otra medida de similitud es la distancia de Levenshtein, la cual se define como la mínima cantidad de cambios de eliminar, cambiar y agregar que se tienen que hacer a dos secuencias para que sean iguales. Esta medida se puede usar tanto en palabras, en donde se mediría la cantidad de cambios a los caracteres, como en listas de palabras, donde se mediría la cantidad de palabras que se tienen que cambiar.

Curvas de aprendizaje

Es necesario además de evaluar el resultado final del modelo, evaluar el proceso de entrenamiento. En esta etapa se pueden diagnosticar varias deficiencias en este proceso. Para un correcto entrenamiento se divide el conjunto de datos en tres partes:

- **entrenamiento**: utilizada para el entrenamiento del modelo.
- **validación**: utilizada para evaluar el desempeño del modelo durante el entrenamiento.
- **prueba**: utilizada para evaluar el resultado final.

Las curvas de aprendizaje constituyen la principal herramienta para evaluar el proceso de aprendizaje. Estas están formadas por las mediciones de métricas a lo largo del entrenamiento calculadas a partir de los conjuntos de validación y entrenamiento. En estas, el eje horizontal representa el número de época del entrenamiento, iteración sobre el conjunto de entrenamiento, y el eje vertical representa el valor de la métrica a analizar. La línea correspondiente al conjunto de entrenamiento cuantifica el aprendizaje del modelo o también el error de entrenamiento, y la correspondiente a la de validación cuantifica la generalización o el error de generalización. Existen tres comportamientos esenciales a analizar:

- Bajo ajuste (*underfitting*).

- Sobreajuste (*overfitting*).
- Buen ajuste.

El bajo ajuste ocurre cuando el modelo no es capaz de aprender del conjunto de datos o cuando este aún puede aprender más. Las curvas de aprendizaje en estos casos se caracterizan por ser una línea plana o valores ruidosos con alta pérdida (Figura 2.6).

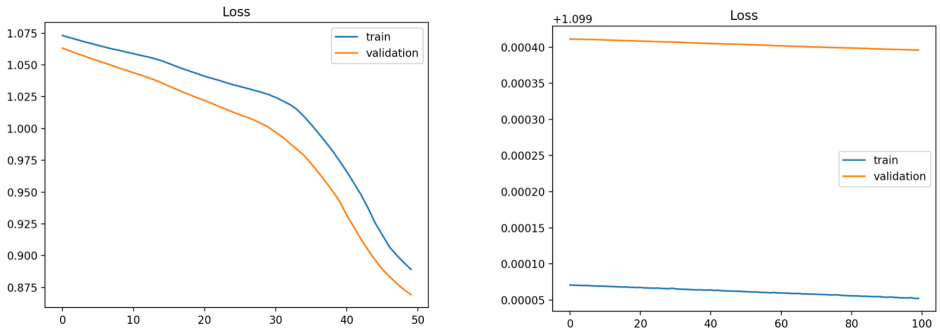

Figura 2.6: Curvas de entrenamiento con bajo ajuste por falta de entrenamiento (izquierda) y por modelo que no aprende de los datos (derecha) (Tomado de Brownlee (2018) pág. XXVII).

Entre las formas más sencillas de combatir el bajo ajuste de los modelos consiste en complejizarlo, al añadir capas o aumentar las dimensiones de este aumenta su expresividad y, por lo tanto, su ajuste. Si este método no funciona es posible considerar un cambio de arquitectura hacia una que pueda extraer más información de la estructura de los datos.

El sobreajuste es el fenómeno en el que el modelo aprende los datos de entrenamiento extremadamente bien, incluso el ruido en estos, esto trae consigo que falla en generalizar el problema para nuevas entradas. Las curvas características de este fenómeno presentan una divergencia en los errores de entrenamiento y validación a medida que se entrena el modelo, mientras que la de entrenamiento mejora la de validación tiende a empeorar (Figura 2.7).

Existen varios métodos para combatir el sobreajuste, uno sencillo es simplificar el modelo quitándole capas o disminuyendo sus dimensiones. Además de esto, existen regularizaciones que se pueden aplicar para evitar que las capas dependan exclusivamente de pocos atributos, entre esta familia los más usados son la regularización L1 y L2 las cuales se definen como la suma del valor absoluto de los atributos y la suma del cuadrado de sus atributos respectivamente. Otra medida para prevenir el sobreajuste

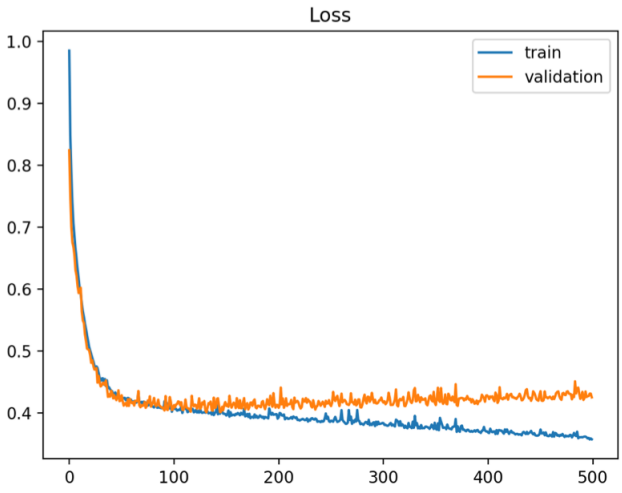

Figura 2.7: Curvas de entrenamiento con sobreajuste (Tomado de Brownlee (2018) pág. XXIX).

es el agrego de capas de abandono (*dropout*). Estas capas desactivan neuronas de la arquitectura, obligando a estas a ser robustas y depender del comportamiento de la población, en lugar de la actividad de otras unidades específicas [Baldi y Sadowski 2013]. La terminación temprana (*early stopping*) del entrenamiento se utiliza para parar este en el momento en que el error de generalización comienza a subir, impidiendo así que se sobreentrene el modelo.

Finalmente, un buen ajuste es el resultado que se alcanza cuando tanto la curva de validación como de entrenamiento presentan valores pequeños y similares, consecuentes con una correcto aprendizaje y generalización (Figura 2.8).

Otro problema observable a partir del análisis de las curvas de aprendizaje constituye la detección de conjuntos de datos no representativos. Un conjunto de datos no representativo es uno que puede no capturar las características estadísticas relativas a otro conjunto de datos extraído del mismo dominio. Esto puede pasar que los conjuntos de entrenamiento o de validación. En caso del conjunto de entrenamiento se puede identificar si la pérdida en el conjunto de entrenamiento conlleva a una ganancia en el conjunto de validación y viceversa quedando al final con una separación entre ambos valores. En el caso del conjunto de validación se presenta como una curva ruidosa, también se puede dar el caso de que el conjunto de validación sea más fácil de predecir que el de entrenamiento, en este caso se observa como la curva de validación permanece siempre por debajo de la de entrenamiento (Figura 2.9).

Para combatir estos problemas se puede aumentar la cantidad de elementos en los conjuntos de entrenamiento o validación en dependencia de donde ocurra.

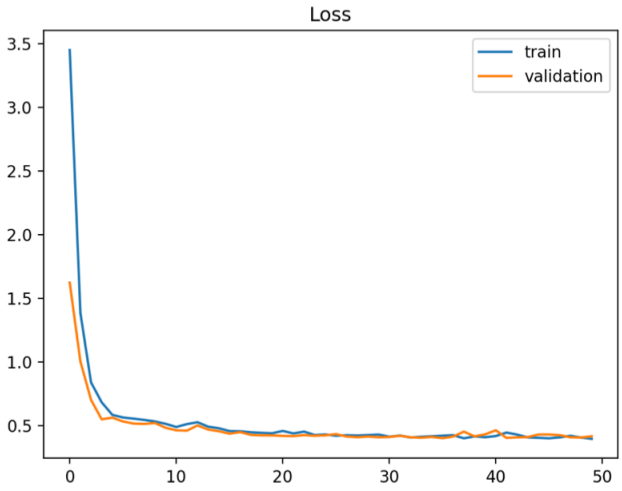

Figura 2.8: Curvas de entrenamiento con buen ajuste (Tomado de Brownlee (2018) pág. XXX).

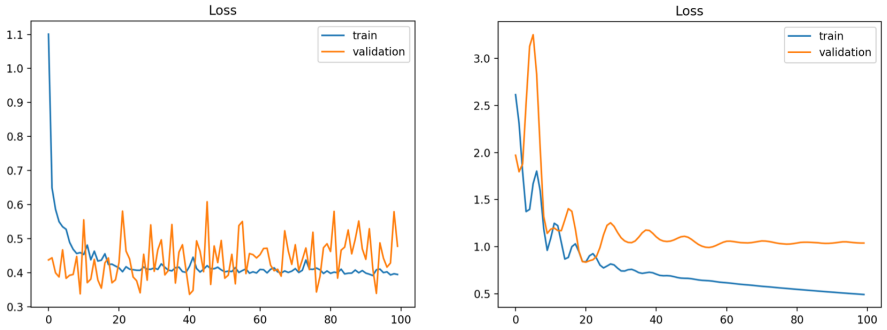

Figura 2.9: Curvas de entrenamiento con datos poco representativos en el conjunto de validación (izquierda) y en el conjunto de entrenamiento (derecha) (Tomado de Brownlee (2018) pág. XXXII).

2.1.5. Aumento de datos

El aumento de datos consiste en acciones para aumentar la diversidad de un conjunto de datos sin recolectar nuevos datos explícitamente [Feng y col. 2021]. En datos continuos, como imágenes o valores numéricos el aumento de datos puede ser realizado al añadirle perturbaciones a entradas existentes, en caso de las imágenes técnicas como el volteado (*flipping*) o recortado (*cropping*) son usadas. Los textos

son un tipo de datos discreto y, por lo tanto, las técnicas anteriores no pueden ser aplicadas directamente. Para el PLN se han estudiado diversas técnicas de aumento de datos, una de estas consiste en el cambio del árbol de dependencia de la oración mediante operaciones de intercambio y borrado de nodos [Şahin y Steedman 2019], también se han utilizado el intercambio de palabras por sinónimos [Dai y Adel 2020] y la traducción de textos hacia un lenguaje y luego de vuelta al lenguaje origen (*backtranslation*) [Sennrich, Haddow y Birch 2015].

2.1.6. Aprendizaje Conjunto

El aprendizaje conjunto (*ensemble learning*) son técnicas encaminadas al aprovechamiento de soluciones encontradas por diferentes modelos, combinándolas y mejorándolas para encontrar una mejor solución al problema. Estos métodos son efectivos en la reducción de la varianza y el sesgo de los modelos, obteniendo así mejores resultados [Dietterich 2002]. Una forma de este tipo de aprendizaje en problemas de clasificación constituye el voto conjunto, en donde los diferentes clasificadores votan sobre la clase a la que pertenece el elemento y, al final, se asigna la etiqueta que más votos obtuvo.

2.1.7. Métodos de optimización

El objetivo del AA es encontrar los extremos de una función de costo, este proceso es una tarea desafiante ya que la gran mayoría de estas funciones no son convexas y, por lo tanto, no existe un algoritmo que asegure la convergencia hacia un extremo global. Para resolver este problema existen múltiples heurísticas, la más usada es el descenso por gradiente. La idea básica consiste en el cálculo del vector gradiente de la función de error f con respecto a los parámetros del modelo x y, una vez se tiene dicho vector, se evalúa en la asignación actual de los parámetros x_i y se realiza un corrimiento de este punto en contra del gradiente para disminuir el error (Eq. 2.22).

$$x_{i+1} = x_i - \alpha \nabla f(x_i) \tag{2.22}$$

En la ecuación 2.22 anterior, α es la tasa de aprendizaje (*learning rate*), que cuantifica cuánto se toma del vector de gradiente para actualizar los parámetros, esto se puede ver como el aprendizaje del modelo.

Variantes eficientes de este algoritmo para el entrenamiento de modelos de AA han sido creadas, las variaciones se encuentran principalmente en la selección de α en cada paso y la selección de los conjuntos de datos con que se estimará el gradiente. Entre las técnicas utilizadas se encuentran Descenso por Gradiente Estocástico, variaciones de tasa de aprendizaje dinámica con sus diferentes variantes (exponencial, polinómica), RMSProp [Tieleman e Hilton 2012] y Adam [Kingma y Ba 2014].

2.2. Preliminares de Extracción de Argumentos

Varias investigaciones han dado respuesta a los problemas asociados a EA, mostrando una variedad en enfoques y métodos.

En Palau y Moens (2009) se propone el uso de modelos estadísticos como *Naive Bayes* (NB) y *Support Vector Machine* (SVM) para la clasificación de oraciones en argumentativas o no y en su rol argumentativo en caso de que sea argumentativa. En este se asume que las componentes argumentativas son oraciones completas. Para la predicción de relaciones se usa un enfoque basado en reglas con la creación de una Gramática Libre de Contexto. Las representaciones de las oraciones consisten en atributos creados a mano, dado el conocimiento experto sobre la argumentación en el tema tratado, elementos como adverbios, verbos, signos de puntuación, palabras clave, estadísticas del texto (tamaño de oración, distancia media de palabras) son usados para la extracción y clasificación de las UDAs, además, se usan también como base en la creación de las reglas de la gramática para la extracción de relaciones.

Goudas y col. (2015) al igual que Palau y Moens (2009) clasifica a las oraciones como argumentativas o no, mediante diferentes clasificadores como NB, *Random Forest*, Regresión Logística y SVM. Sin embargo, Goudas y col. (2015) aumenta la granularidad de la segmentación al permitir la extracción de los segmentos que contienen la carga argumentativa de dentro de las oraciones previamente clasificadas como tal, esto se realiza mediante la extracción de etiquetas BIO de las oraciones con el uso de un CRF. La predicción de las relaciones es modelado como un problema de clasificación usando SVM para clasificar pares de UDAs en relacionados o no. Atributos creados a mano son usados en la extracción de UDAs; entre estos están posición de la oración en el texto, cantidad de verbos, comas, adverbios, palabras, entidades en la oración, también se emplean listas que guardan entidades relacionadas con el dominio específico y palabras clave indicadoras de frases argumentativas.

Stab y Gurevych (2017) proponen un mecanismo de segmentación basado en CRF. La clasificación y predicción de relaciones se modela conjuntamente con dos clasificadores SVM y un problema de Optimización Lineal Entero que encuentra la mejor estructura y asegura una disposición arbórea. En la segmentación de las UDAs, se extraen por cada token su posición en el texto, si precede o sucede a un signo de puntuación, su parte de la oración, la probabilidad de que sea el comienzo de una UDA dado sus tokens anteriores, entre otros. Para la extracción y clasificación de relaciones se proponen otros conjuntos de atributos como la cantidad de sustantivos comunes entre las componentes fuente y el objetivo, la presencia de indicadores argumentativos, representaciones vectoriales de tokens, entre otros.

En Eger, Daxenberger y Gurevych (2017) trabajan el problema de EA como uno *end-to-end*. Para esto presentaron varias propuestas, entre ellas se encontraba modelar el problema como uno de secuencia a secuencia, usando RNN como LSTM en versiones

bidireccionales capturando información desde ambos lados de la secuencia. Para la representación de las palabras se extrajo información morfológica de las palabras mediante la aplicación de una CNN a los caracteres de estas, al final, realizan la clasificación de la secuencia con un CRF. Realizaron experimentos al modelar el problema como uno de *Dependency Parsing* [Kiperwasser y Goldberg 2016]. Este problema consiste en construir un árbol de dependencia que codifique las estructuras argumentativas. En este se tiene que decidir entre varias opciones (*shift*, *reduce*) en dependencia del contenido de la pila y del *buffer* para la confección del árbol. El problema fue modelado también como un problema de reconocimiento de entidades nombradas, en donde las entidades son las UDAs.

En Dykes y col. (2020) se proponen métodos basados en reglas para la extracción de argumentos sobre textos en Twitter. Estos métodos se centran en la confección de reglas basadas en anotaciones lingüísticas como partes de la oración y lemas de palabras. La recuperación está basada en los esquemas argumentativos comunes presentes en los textos. Dada las reglas creadas y el tipo de datos con que se trabaja, o sea, cadenas de texto pequeñas; estos algoritmos tienden a tener una alta precisión aunque bajo recobrado, esto no es un gran problema en conjuntos de datos grandes, pero en conjuntos de menor tamaño o estructura más compleja pierden efectividad.

Galassi (2021) propone el uso de redes residuales y mecanismos de atención para la creación de un modelo que, conjuntamente, clasifica el tipo de UDA y la relación existente entre estas. Este trabajo define el concepto de distancia argumentativa, añadiéndolo como característica y asume que las UDAs ya fueron extraídas. En este caso, además de la distancia argumentativa, las secuencias son representadas vectorialmente con GloVe.

En resumen, se contemplan disímiles enfoques al problema de EA desde una perspectiva enmarcada en modelos simbólicos, estadísticos y neuronales en versiones tanto secuenciales como *end-to-end*. Cada uno de estos modelos presentan sus ventajas y desventajas a la hora de construirlos, extenderlos y comprender su funcionamiento. En modelos simbólicos se presenta una alta precisión en dominios específicos debido a que se construyen teniendo en cuenta reglas específicas a un contexto dado. Estos modelos son poco escalables y difíciles de mantener ya que sus reglas son construídas a mano y dicho proceso requiere de conocimiento experto y tiempo. Los modelos estadísticos se caracterizan por usar conjuntos de atributos creados a mano, dichos atributos son difíciles de encontrar, calcular y pueden no poseer relevancia en otros contextos diferentes a los que fueron creados, además, la necesidad de conocimiento experto es necesaria para su confección. Los modelos neuronales poseen una mayor adaptabilidad, en estos la entrada puede ser codificada en una representación que es aprendida por el mismo algoritmo, permitiendo su uso en esquemas argumentativos con características diferentes. Los modelos simbólicos y estadísticos poseen la ventaja de poder explicar el porqué de los resultados devueltos cosa que se vuelve casi

imposible en modelos neuronales.

Dado que la EA es un proceso en el cual se necesita pasar por varias tareas, estas deben de ser completadas de alguna forma. Una manera de completarlas es hacerla una a la vez, independiente una de otra y pasándole la salida de etapas anteriores a las etapas siguientes. Esta manera secuencial de realizar las tareas es bastante simple y ayuda a la creación de modelos simples y con tareas bien definidas, aunque trae consigo la propagación de los errores a través del proceso y el no aprovechamiento de las interrelaciones entre variables computadas de procesos anteriores. También requiere de la construcción, entrenamiento y evaluación de varios modelos. En cambio un enfoque *end-to-end* poseen la habilidad de modelar el problema desde su inicio hasta su final de manera conjunta, mediante *Multi-Task Learning* (MTL) se modelan las tareas de manera conjunta creando un solo modelo complejo con una propagación de error menor.

2.3. Proyección de corpus

La EA no presenta una gran cantidad de datos anotados con los cuales se pueda realizar un entrenamiento, además de esto la gran mayoría de corpus existentes se encuentran en lenguajes como inglés o alemán, haciendo difícil el desarrollo de esta rama en otros lenguajes. La escasez de estos datos es, en gran parte, debida al elevado costo monetario, de tiempo y de recursos humanos que se utiliza en su creación. En orden de poder desarrollar la EA en otros lenguajes, como el español, se han investigado diferentes vertientes para la construcción de conjuntos de datos en estos lenguajes de pocos recursos, a partir de los conjuntos de datos ya existentes.

La proyección de etiquetas consiste en un algoritmo en donde se transfieren las etiquetas de un corpus anotado a nivel de tokens en un lenguaje origen hacia su traducción en un lenguaje objetivo. En Eger, Daxenberger, Stab y col. (2018) se propone un algoritmo de proyección dadas las alineaciones de palabras. El proceso se divide en varias partes:

1. Traducción de oraciones.
2. Alineación de palabras.
3. Proyección de etiquetas.

2.3.1. Traducción de oraciones

La Traducción Automática consiste en el proceso de usar inteligencia artificial para traducir texto de un lenguaje fuente a un lenguaje objetivo sin la intervención humana. En la actualidad, este campo ha dado un gran paso pasando de modelos

estadísticos a modelos neuronales obteniendo traducciones de una alta calidad sin variar significativamente de la humana, condición necesaria para una buena proyección [Eger, Daxenberger, Stab y col. 2018].

Este primer paso de la proyección de corpus consiste en traducir todas las oraciones existentes en el conjunto de datos hacia el lenguaje objetivo. A continuación se muestra una oración en inglés y su traducción al español[2]:

Firstly , people normally have lots of things to do .
En primer lugar , la gente normalmente tiene muchas cosas que hacer .

2.3.2. Alineación de palabras

La alineación de palabras consiste en encontrar las palabras generadas en el lenguaje objetivo por las palabras en el lenguaje fuente. Algoritmos basado en modelos bayesianos, como FastAlign [Dyer, Chahuneau y Smith 2013], y Cadenas de Markov-Monte Carlo, como EFEMARAL [Östling y Tiedemann 2016] se ubican entre las primeras herramientas para la solución del problema. Modelos más recientes se han enfocado en explotar las representaciones vectoriales de palabras y el uso de métodos de atención para la extracción de las alineaciones [Dou y Neubig 2021]. Algunas consideraciones sobre el proceso: las relaciones formadas entre palabras pueden ser de tipo muchos a muchos, además de no tener el mismo orden de la oración inicial o incluso no estar relacionadas directamente con una palabra en la oración objetivo. Estas consideraciones dan una medida de la dificultad de la tarea en cuestión. En el ejemplo siguiente se observa el resultado de las herramientas de alineación, en este las palabras en el idioma de origen (inglés) están anotadas con su posición en la oración y las palabras en el idioma objetivo (español) están anotadas con la posición de la palabra que la originó en el idioma origen:

Firstly_0 $,_1$ people_2 normally_3 have_4 lots_5 of_6 things_7 to_8 do_9 $.{}_{10}$
En primer_0 lugar_0 $,_1$ la gente_2 normalmente_3 tiene_4 muchas_5 cosas_7 que_8 hacer_9 $.{}_{10}$

2.3.3. Proyección de etiquetas

La proyección de etiquetas consiste en transportar las etiquetas de las palabras en la secuencia origen hacia las palabras de la secuencia destino tomando como datos las alineaciones entre estas. En [Yarowsky y Ngai 2001] se trata el problema de proyección de frases nominales, estas frases tienen como característica que son resistentes a ser divididas en caso de ser traducidas, y aunque evidencian cambios en el orden de las palabras, mantienen la misma ventana; dicha propiedad se cumple para las UDAs

[2]Extraído del corpus de Stab y Gurevych (2017).

también. La proyección de UDAs es más simple en dado que solamente se tiene en cuenta la ventana y las etiquetas en estas son constantes, no pasa con la proyección en frases nominales, las cuales pueden cambiar dentro de una ventana, por lo que algoritmos más simples existen para esta tarea [Eger, Daxenberger, Stab y col. 2018]. En el ejemplo de proyección están anotadas las etiquetas originales en formato BIO de las palabras de la oración en el lenguaje origen (inglés) y se muestra el resultado de proyectar estas al lenguaje objetivo utilizando los resultados de la alineación de palabras:

> Firstly$_O$,$_O$ people$_B$ normally$_I$ have$_I$ lots$_I$ of$_I$ things$_I$ to$_I$ do$_I$.$_O$
> En$_O$ primer$_O$ lugar$_O$,$_O$ la$_O$ gente$_B$ normalmente$_I$ tiene$_I$ muchas$_I$ cosas$_I$ que$_I$ hacer$_I$.$_O$

Capítulo 3

Propuesta

El modelo propuesto se divide en dos secciones. En la primera sección se realiza la segmentación y clasificación de las UDAs como tareas conjuntas. En la segunda sección se predicen los enlaces y sus clasificaciones, tomando como tareas auxiliares la clasificación de las UDAs. Dada la heterogeneidad de las conjuntos de datos disponibles en EA, los modelos poseen un mínimo de atributos, esto permite que el modelo por sí solo aprenda la mejor representación para el esquema de anotación con que se entrene.

3.1. Segmentación y clasificación de UDAs

Esta primera parte se modela como un problema secuencia a secuencia cuyo objetivo es asignar a los tokens extraídos del documento entrada una etiqueta BIOES para segmentar las UDAs. Para la clasificación del tipo de UDA, al conjunto de etiquetas BIES se le añadieron las clasificaciones que presenta el corpus entrenante. En el siguiente ejemplo se muestra una salida del modelo presentando las clasificaciones de A como argumento y P como premisa:

> En_{O} primer_{O} lugar_{O} $,_{O}$ [el_{B-A} $correo_{I-A}$ $electrónico_{I-A}$ $puede_{I-A}$ $contar_{I-A}$ $como_{I-A}$ uno_{I-A} de_{I-A} los_{I-A} $resultados_{I-A}$ $más_{I-A}$ $beneficiosos_{I-A}$ de_{I-A} la_{I-A} $tecnología_{I-A}$ $moderna_{E-A}$] $._{O}$ [$Años_{B-P}$ $atrás_{I-P}$ $,_{I-P}$ las_{I-P} $personas_{I-P}$ $pagaban_{I-P}$ $gran_{I-P}$ $cantidad_{I-P}$ de_{I-P} $dinero_{I-P}$ $para_{I-P}$ $enviar_{I-P}$ sus_{I-P} $cartas_{I-P}$ y_{I-P} sus_{I-P} $pagos_{I-P}$ $estaban_{I-P}$ $sujetos_{I-P}$ al_{I-P} $peso_{I-P}$ de_{I-P} sus_{I-P} $cartas_{I-P}$ o_{I-P} $paquetes_{I-P}$ y_{I-P} $muchos_{I-P}$ $accidentes_{I-P}$ $podrían_{I-P}$ $causar_{I-P}$ $problemas_{I-P}$ que_{I-P} $causarían_{I-P}$ que_{I-P} el_{I-P} $correo_{I-P}$ no_{I-P} $fuera_{I-P}$ $enviado_{E-P}$] $._{O}$

3.1.1. Modelo de segmentación y clasificación de UDAs

Sea D un documento entrada, este es separado en una secuencia de n tokens D_i donde n es la mayor longitud encontrada en los documentos del conjunto de datos (si la cantidad de tokens es menor que n entonces D_i es completado con un token especial de enmascarado). A cada token se le asigna su representación vectorial GloVe de dimensión $g = 300$, dando como resultado $G_{ij} \in \mathbb{R}^{n \times g}$. Esta representación inicial presenta información semántica de las palabras y conserva las relaciones espaciales entre ellas.

Para la representación de información morfológica de la palabra se construyen dos codificadores que procesan los caracteres de cada token y devuelven una representación vectorial de estos. A cada caracter se le asigna un vector que será entrenado convirtiendo un token en un vector de dimensión $q \times c$, donde q es el tamaño máximo de palabra en el conjunto de datos y c es la dimensión del vector asignado a cada caracter. Uno de estos modelos está basado en CNN, este modelo entrena una representación de caracteres de dimensión $cd = 50$ representando un token como un vector de dimensión $q \times cd$. Se conforma por una capa de convolución unidimensional con $f = 30$ filtros y un kernel de tamaño $k = 3$, seguida por una capa *max pooling* que convierte la secuencia en un vector de dimensión $1 \times f$, que luego es concatenado a la representación del token a que pertenece. Otro modelo utilizado para calcular una representación morfológica se encuentra basado en RNN. Se usó un modelo LSTM bidireccional con dimensión $l = 25$ para calcular la representación del token, para las dimensiones de los caracteres se utilizan vectores de tamaño l, el resultado final constituye la concatenación de la corrida hacia adelante y hacia atrás formando una representación de dimensión $1 \times 2 \cdot l$ del token. Este vector es concatenado a la representación del token correspondiente. Otro atributo usado en la representación de los tokens constituyen las etiquetas de Partes de la Oración de estos. El conjunto de etiquetas elegido es un conjunto universal [Petrov, Das y McDonald 2011] aplicable a cualquier idioma. De estas etiquetas se les extrae la codificación *one-hot* y esta es transformada por una capa densa con $p = 5$ neuronas y función de activación $ReLU$, el resultado es concatenado a la representación del token correspondiente. Mediante la extracción de estos atributos el token es representado en tres maneras, semántica, morfológica y estructural, con el objetivo de que sean aprendidas los rasgos lingüísticos correspondientes a la tarea.

Del proceso de vectorización sale un vector con dimensión $n \times t$ donde t es la dimensión final de la representación de los tokens. Este vector es modificado por una capa LSTM bidireccional de dimensión $m = 200$, a esta salida se le añade una conexión residual al ajustarle la dimensión con una capa densa. Luego, la secuencia es procesada por una capa densa de dimensión $k = 100$ con activación $ReLU$ produciendo una representación final de dimensión $n \times k$. Finalmente, se utiliza una capa CRF para la clasificación final de la secuencia en las etiquetas finales. El resultado final

constituye un vector de dimensión n que representa las clasificaciones inferidas por el modelo (Figura 3.1).

Para prevenir el sobreajuste se agregaron capas de normalización y de *dropout* entre cada proceso y se usaron regularizaciones L2 y *dropout* en las capas densas y LSTM, el valor asignado al dropout es de 0,5. Para prevenir el sobreentrenamiento se aplicó una terminación temprana de este cuando no se encontraba una mejora de la función de pérdida en el conjunto de validación por más de 10 épocas consecutivas. Como optimizador se utilizó Adam con una tasa de aprendizaje de 0,001.

3.1.2. Posprocesamiento de segmentación y clasificación de UDAs

La salida del modelo constituye una secuencia de etiquetas en formato BIOES. Esta está propensa a contener errores en su formato, por ejemplo, secuencias no terminadas en E, segmentos continuos con más de una meta-etiqueta, entre otros. Para la corrección de la estructura se propone el siguiente algoritmo con dos partes. La primera consiste en arreglar la estructura BIOES, para esto se mantiene una ventana de tamaño 3, [_ , _ , _], sobre la secuencia y se asume que la parte anterior a la posición de la ventana no presenta errores. Al encontrar una ventana inválida se necesita observar la siguiente ventana para poder decidir cómo se arregla el error, ya que se podría dar el caso que se observe [O, O, I] y la próxima sea [O, I, O], en donde solamente viendo la primera ventana no se podría saber si el cambio correcto corresponde a sustituir I por B o por S. Una vez observadas las dos ventanas, se procede a realizar el arreglo correspondiente. En casos donde sea ambigua la manera de arreglar la ventana, [I, I, O] por ejemplo (La I o la O pueden ser sustituidas por una E), se utiliza una función que recibe un segmento y devuelve la gravedad del error. El error con mayor gravedad será arreglado, en caso de ser iguales se arreglará la etiqueta más a la izquierda. Este procedimiento devuelve una secuencia BIOES correctamente anotada, debido a que a partir de una secuencia sin errores en cada paso se va arreglando la ventana y una vez esta llega al final arregló todos los elementos de la secuencia. Una vez la secuencia tiene la estructura BIOES correctamente anotada el problema consiste en arreglar las meta-etiquetas, ya que una misma secuencia BIOES pudo haber sido anotada con diferentes tipos, en este caso se toma la etiqueta más representativa del segmento continuo.

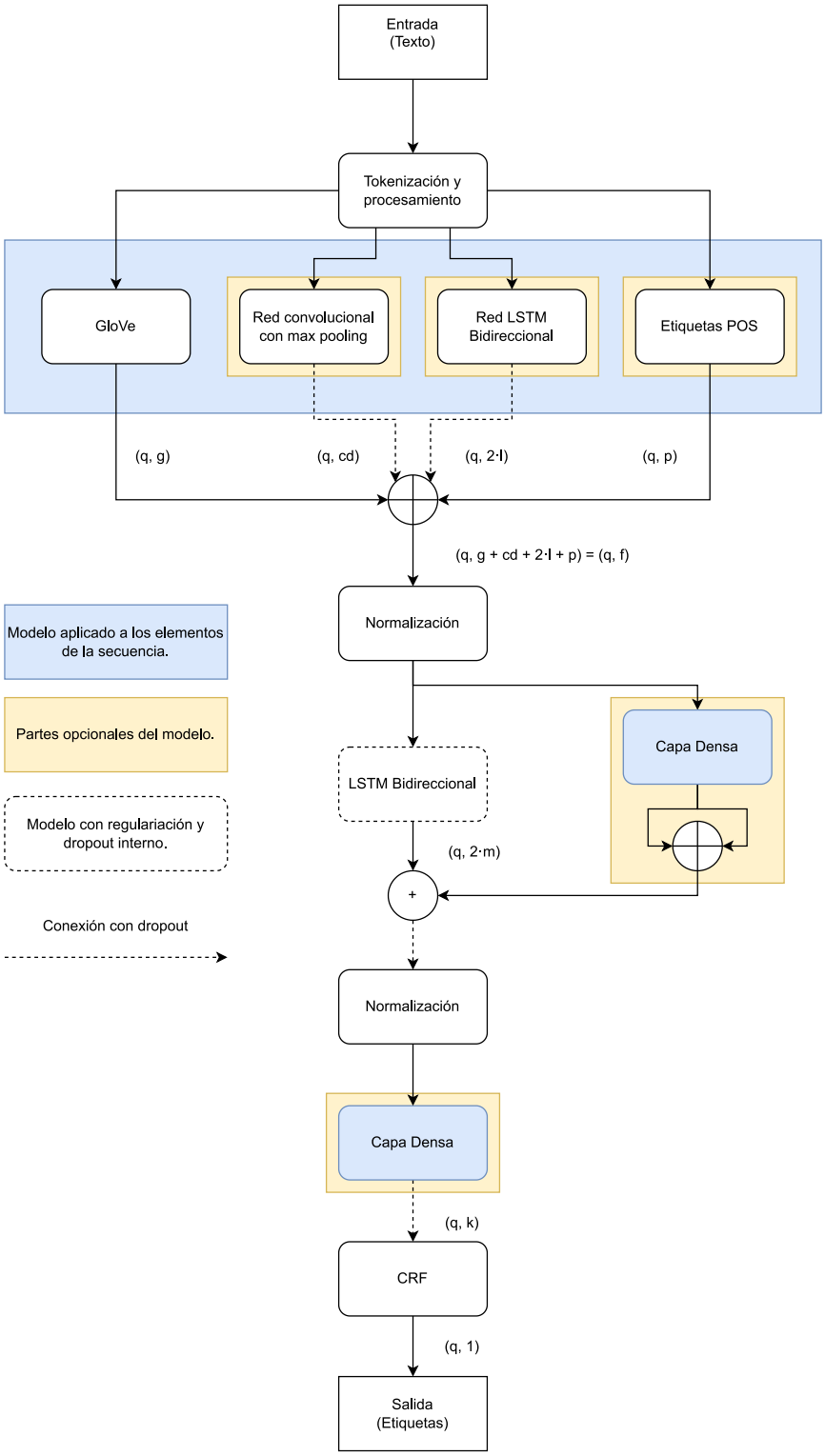

Figura 3.1: Segmentador UDAs.

3.2. Predicción y clasificación de enlaces

En esta segunda parte el modelo se encarga de las tareas de extracción y clasificación de enlaces. El problema consiste en clasificar pares de UDAs, representando origen y objetivo del enlace, en el tipo de relación que existen entre estas. Como tarea auxiliar se clasifican los tipos de UDAs que intervienen en la relación. La salida del modelo constituye en una tupla de tres elementos: la clasificación de la relación, la clasificación de la UDA origen, la clasificación de la UDA objetivo. Si el enlace existe o no es calculado partir de la clasificación de la relación.

3.2.1. Modelo de predicción y clasificación de enlaces

Sean dos UDAs, S y T, donde S representa la fuente de la relación, mientras que T representa al objetivo. Estas secuencias son tokenizadas y se les asigna la representación GloVe de cada palabra, obteniendo dos vectores de dimensión $u \times g$, donde u es el tamaño máximo de UDAs en el conjunto de entrenamiento y $g = 300$ es la dimensión del *embedding*. Estos vectores son modificados por una red densa compuesta por $ca = 4$ capas con activación *ReLu* de dimensiones 50, 50, 50, 300, añadiendo una conexión residual a la salida de esta. El próximo paso consiste en aplicar una capa densa de dimensión $di = 50$ y luego un *average pooling* de tamaño $dp = 10$, obteniendo vectores de dimensión $\frac{g}{dp} \times di$. Estos vectores son modificados por un LSTM bidireccional con $lm = 50$ unidades. Un módulo de atención es aplicado sobre los vectores fuentes, en este actúan como consultas el promedio de los vectores objetivo y como llaves y valores los vectores fuentes, el procedimiento simétrico es realizado para los vectores objetivos. La salida de los procesamientos son concatenados con la distancia argumentativa obteniendo una representación conjunta de la relación a analizar. Esta representación conjunta es modificada por una red residual obteniendo una representación final de dimensión $l = 20$ y luego sometida a los clasificadores de relación y de tipos de UDAs (Figuras 3.2 y 3.3).

Para prevenir el sobreajuste se agregaron capas de normalización y de *dropout* entre cada proceso y se usaron regularizaciones L2 y *dropout* en las capas densas y LSTM, todos los *dropout* tienen valor $dr = 0,1$. Para prevenir el sobreentrenamiento se aplicó una terminación temprana de este cuando no se encontraba una mejora de la función de pérdida en el conjunto de validación durante $v = 5$ épocas consecutivas. Como optimizador se utilizó Adam con descenso exponencial con tasa de aprendizaje $lr = 0{,}003$.

Dado que se realiza un aprendizaje de varias tareas, se tienen varias funciones de pérdida individuales que conforman la función de pérdida final e. Sea e_r la función de pérdida de la clasificación de la relación, e_s la del tipo de UDA origen y e_t del tipo de UDA objetivo, entonces $e = 10 \cdot e_r + e_s + e_t$.

3.2.2. Preprocesamiento de predicción y clasificación de enlaces

El uso de este modelo se concreta a nivel de documento, en donde ya se tienen las UDAs extraídas. Para alimentar al modelo con los pares de UDAs y sus distancias argumentativas se seleccionan todos los pares de estos que cumplan que no se enlacen con ellos mismos, por ejemplo, $a \rightarrow a$; y que su distancia argumentativa absoluta sea menor que $da = 10$. Estas restricciones disminuyen el número de pares extraídos por documentos a una cantidad lineal con respecto a la cantidad de UDAs, presentes ya que por cada UDA solamente se tomarían $2 \cdot da$ elementos como máximo (los que la preceden y los que la suceden).

Además de las etiquetas R originales del conjunto de entrenamiento, se añaden elementos extras a este conjunto. Estos elementos son las representaciones inversas de las relaciones, por ejemplo, si $a \xrightarrow{c} b$ entonces se agregará el par $b \xrightarrow{c^{-1}} a$, donde c^{-1} es una nueva clasificación de relación que representa el inverso de la clasificación c. Este proceso se realiza para aumentar la cantidad de relaciones positivas en el conjunto entrenante, ya que aun con las reducciones hechas existe un desbalance de clases positivas y negativas en las relaciones.

3.2.3. Posprocesamiento de predicción y clasificación de enlaces

Se calcula una salida extra a partir de las distribuciones de probabilidad de las relaciones devueltas por el modelo, esta salida representa si el par está enlazado o no directamente, para este cálculo se suman las categorías vinculadas a las clases originales del conjunto de datos, esta se toma como la probabilidad de estar enlazados, que en caso de ser mayor del 50 % se devuelve verdadero. Para dar el resultado final se eliminan del conjunto de respuesta las relaciones anotadas con las etiquetas inversas añadidas en el paso de preprocesamiento y son devueltas aquellas que se clasifiquen como enlazadas según el criterio anterior.

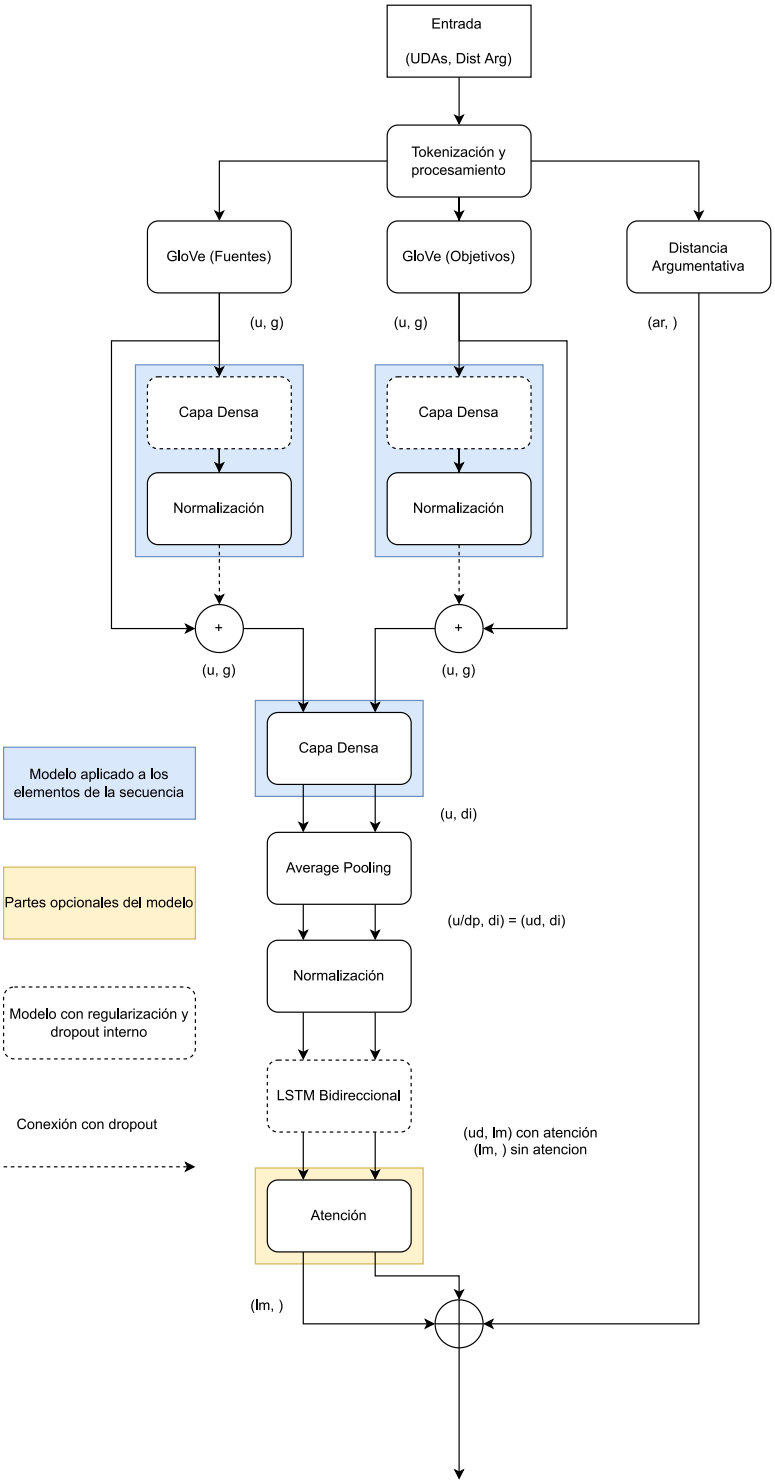

Figura 3.2: Predictor de enlaces.

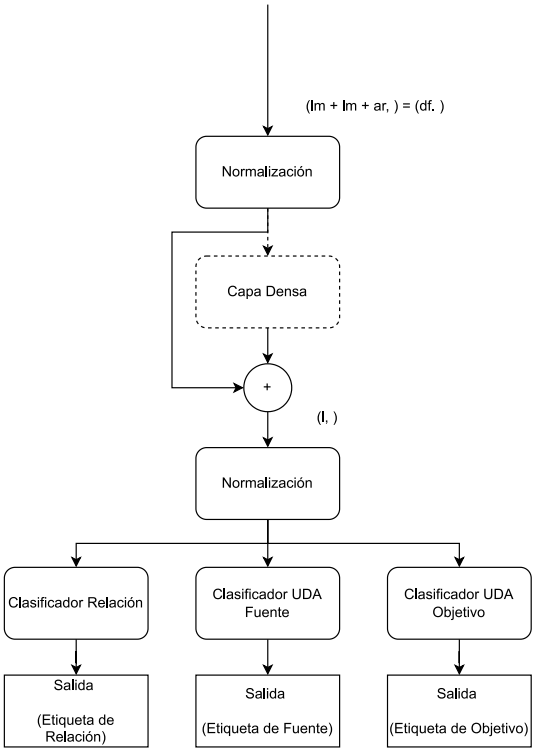

Figura 3.3: Predictor de enlaces (continuación).

Capítulo 4

Experimentación

En el capítulo se describen los conjuntos de datos utilizados para el entrenamiento y validación del modelo propuesto y cómo se construyen los conjuntos de datos en español. Se mencionan las herramientas utilizadas para la confección del software. Además, se describe el proceso de experimentación y evaluación de los modelos construidos con los conjuntos de datos. Finalmente, se presentan los resultados obtenidos al aplicarles los modelos a las Cartas a la Dirección extraídas del periódico Granma.

4.1. Conjuntos de Datos

Para el entrenamiento de los modelos propuestos se utilizaron corpus diferentes, estos presentan esquemas de anotación distintos entre sí, difiriendo principalmente en la definición de UDA y las clasificaciones dadas a estas y a las relaciones. A todos los conjuntos se les proyectó al lenguaje español y se les realizó un aumento de datos. Como conjunto para la validación fueron usadas una parte de las Cartas a la Dirección del periódico Granma.

4.1.1. Ensayos Argumentativos

Este corpus [Stab y Gurevych 2017] presenta unos 402 documentos, dividido por los autores en 286 documentos para entrenamiento (70 %), 80 para prueba (20 %) y 36 para validación (10 %). Los contenidos de estos son ensayos de estudiantes en los que se argumentan sobre temas como cooperar o competir y contribuciones de la tecnología a la sociedad. Las anotaciones de las UDAs se conforman por segmentos de textos argumentativos, estos segmentos son clasificados en *MajorClaim* (751, 12 %), *Claim* (1506, 25 %) y *Premise* (3832, 63 %). La estructura de las relaciones entre las UDAs conforman árboles en los que se tienen como raíz las *MajorClaim* del texto. Las relaciones solo están permitidas entre *Premise-Premise* y *Premise-Claim*, clasificadas

en *attack* (219, 6 %) y *support* (3613, 94 %). Las relaciones entre *Claim* y *MajorClaim* son anotadas de manera diferente, por medio de darle a las *Claim* una clasificación de si está a favor (1228) o en contra (278) de las *MajorClaim* del documento. Para el análisis de este corpus se consideraron las relaciones *Claim-MajorClaim* de igual manera que las otras, al convertir estas posiciones en relaciones. El número final con la inclusión de estas aumenta a 715 (10 %) de ataque y 5958 (90 %) de apoyo.

4.1.2. CDCP

El corpus [Niculae, Park y Cardie 2017] está conformado por 731 comentarios de usuarios extraídos de la web bajo el tema de prácticas de cobro de deudas a los consumidores (CDCP en inglés). Las UDAs se encuentran segmentadas en oraciones y todas se consideran argumentativas, estas son clasificadas en *policy* (815, 17 %), *value* (2180, 45 %), *fact* (785, 16 %), *testimony* (1116, 21 %) y *reference* (32 1 %). Las relaciones se encuentran clasificadas en *reason* (1352, 97 %) y *evidence* (73, 3 %).

4.1.3. AbsTRCT

AbsTRCT [Mayer, Cabrio y Villata 2020] se compone de 500 documentos sobre el estudio de cuatro enfermedades diferentes, glaucoma, hipertensión, hepatitis b y diabetes. Las UDAs constituyen oraciones, aunque no todas son consideradas argumentativas. Estas se clasifican en *MajorClaim* (93, 3 %), *Claim* (993, 30 %) y *Premise* (2198, 67 %). Las relaciones están representadas por tres categorías: *support* (1763, 85 %), *partial-attack* (238, 12 %) y *attack* (60, 3 %).

En resumen, estos datos no son grandes y contiene una gran cantidad de desbalance en sus clases. En la Tabla 4.1 se muestran los datos promedio de la composición de los diferentes corpus. Se observa una composición heterogénea entre estos, principalmente CDCP difiere en gran número de los demás.

Tabla 4.1: Información de promedios de los conjuntos de datos.

Corpus	Tokens	Tokens argumentativos	UDAs	Relaciones por UDA
Ensayos Argumentativos	381	68 %	35 %	1,08
CDCP	127	99 %	97 %	0,30
AbsTRCT	371	50 %	47 %	0,63

4.1.4. Creación de corpus en español

Aumento de datos

A los conjuntos de datos a analizar se les realizó un aumento de datos mediante la técnica de *backtranslation*, aplicándole la proyección de etiquetas de los elementos originales a los aumentados. Esto contribuyó a duplicar la cantidad de elementos disponibles. Los resultados obtenidos al comparar los elementos originales con los aumentados se reflejan en la Tabla 4.2. Estos muestran que se logró una variación pequeña en los datos, aunque conservando la longitud original del texto.

Tabla 4.2: Datos promedios comparando los textos originales con los aumentados.

Corpus	Jaccard	Levenshtein	$\frac{\lvert\text{Palabras originales}\rvert}{\lvert\text{Palabras aumentadas}\rvert}$
Ensayos Argumentativos	0.69	96	1.00
CDCP	0.72	30	1.02
AbsTRCT	0.74	86	1.04

Proyección de corpus

Todos los conjuntos de datos están originalmente en inglés, por lo tanto, se les aplicó el algoritmo de proyección de corpus para obtener uno en español para ser usado en el entrenamiento de los modelos. Para la traducción automática se utilizó el servicio de Google Translate[1]. Para calcular las alineaciones de palabras se probaron dos algoritmos: FastAlign [Dyer, Chahuneau y Smith 2013] y AwesomeAlign [Dou y Neubig 2021]. Se observó que el primero, aunque es más rápido posee una calidad menor en los resultados, el segundo posee una mayor calidad, aunque requiere de mayor tiempo y recursos para ejecutarse. Para los experimentos se usó finalmente AwesomeAlign. La proyección de las etiquetas fue llevada a cabo por el algoritmo propuesto en Eger, Daxenberger, Stab y col. (2018).

4.1.5. Cartas a la Dirección

Las Cartas a la Dirección constituyen un segmento del periódico Granma donde son publicadas cartas enviadas por la población o empresas a dicha entidad. En general, las cartas presentan dudas o problemas de la población con el objetivo de obtener respuestas del organismo asociado. Se extrajeron 2891 cartas desde el 30 de agosto del 2013 hasta el 28 de octubre del 2022. Estas contienen aproximadamente 975000

[1]https://translate.google.com/

palabras en los datos, en promedio, la cantidad de palabras por carta es de 330. Se encontraron 874 cartas en respuesta a cartas enviadas, lo que representa un 30 % del total. Se extrajeron los comentarios asociados a las cartas, en este sentido 987 cartas no presentan comentarios y, en promedio, se realizan 2 comentarios por carta. Los textos presentan un título y un formato relativamente libre, aunque en las cartas de respuesta se puede observar una firma de la persona que respondió y la entidad que representa. Del total de cartas, se seleccionaron las que fueran en respuesta a otra y también las cartas que fueron respondidas para tener una mayor concentración de cartas que fueran argumentativas, esta selección está conformada por 1702 cartas, lo que representa un 59 % del total de cartas.

4.2. Implementación

La implementación de los modelos y algoritmos de procesamiento y visualización de datos se encuentran en un repositorio de GitHub[2]. Esta implementación está concebida para que se pueda extender fácilmente para el uso con otros idiomas diferentes del inglés y el español. Se basa en una arquitectura de procesamiento secuencial en el cual cada paso del proceso realiza una tarea específica y lo más desacoplada posible de las otras. Las tareas realizadas son:

- Creación del corpus en un formato estándar: dado que los corpus vienen en diferentes formas, este paso se realiza para trabajar sobre una misma representación de este.
- Proyección del corpus de un lenguaje fuente a un lenguaje objetivo: en el caso de uso del trabajo se proyecta del inglés al español.
  - Traducción y alineación de oraciones.
  - Alineación de palabras.
  - Proyección de etiquetas.
- Extracción y clasificación de UDAs.
- Extracción y clasificación de las relaciones entre las UDAs.
- Visualización de los resultados.

[2] `https://github.com/luisoibarra/argument-mining`

4.2.1. Herramientas

El lenguaje empleado para la confección del software fue Python [*Python* s.f.], este presenta una gran variedad de herramientas para el trabajo con texto, visualización de datos y creación de modelos de aprendizaje profundo. Se utilizó tensorflow [*Tensorflow* s.f.] en su versión 2.9.2 para la construcción y entrenamiento de los modelos. Para el procesamiento de los textos se utilizaron nltk [*Natural Language Toolkit* s.f.] y spacy [*Spacy · Industrial-strength natural language processing in python* s.f.], con estos se realizaron tareas como la extracción de tokens y oraciones del texto, la anotación de las etiquetas de partes de la oración. Se utilizaron ambos paquetes para el procesamiento debido a que, en dependencia de la situación, cada uno presenta diferentes ventajas. En el caso de nltk, presenta algoritmos rápidos para el procesamiento de texto que no requieren de muchos recursos computacionales, sin embargo, estos algoritmos no están disponibles de inmediato para otros lenguajes como el español. Spacy por su parte presenta algoritmos más certeros a costo de mayor tiempo de procesamiento y gasto de recursos computacionales, y también presenta una cantidad mayor de lenguajes disponibles. Para la visualización y manejo de los datos, y cálculo de métricas se utilizaron matplotlib [*Visualization with python* s.f.], pandas [*Pandas* s.f.] y sklearn [*Scikit-Learn* s.f.]. Para la recolección de las Cartas a la Dirección del periódico Granma se utilizó scrapy [*A fast and powerful scraping and web crawling framework* s.f.]. Como interfaz visual para el usuario se utilizó la herramienta Brat [*Brat Rapid Annotation Tool* s.f.] (Figura 4.1). Esta herramienta permite la visualización y edición de las estructuras argumentativas. Dado que Brat es una página web, esta se puede desplegar y permite su uso online.

4.2.2. Formato Estándar

El formato estándar creado es basado en el esquema de anotación CoNLL, donde se anotan a nivel de token todos los aspectos relevantes para las tareas a realizar. La segmentación de las UDAs son representadas por las anotaciones BIO o BIOES en cada palabra, en adición, las clasificaciones de estas son anotadas al adicionar el nombre de esta separada por un guión. Las relaciones son anotadas auxiliándose de la distancia argumentativa, estas son agregadas al anotar el tipo de relación con su respectiva distancia ambas separadas por un guión. A continuación se muestran ejemplos de este formato, conformado por el token, su clasificación BIOES, su clasificación UDA, y las relaciones representadas por su clasificación y su distancia argumentativa:

- Elemento fuera de una UDA: $análisis \quad O$
- UDA intermedia con una relación: $contribuye \quad I - Premise - attacks - 5$

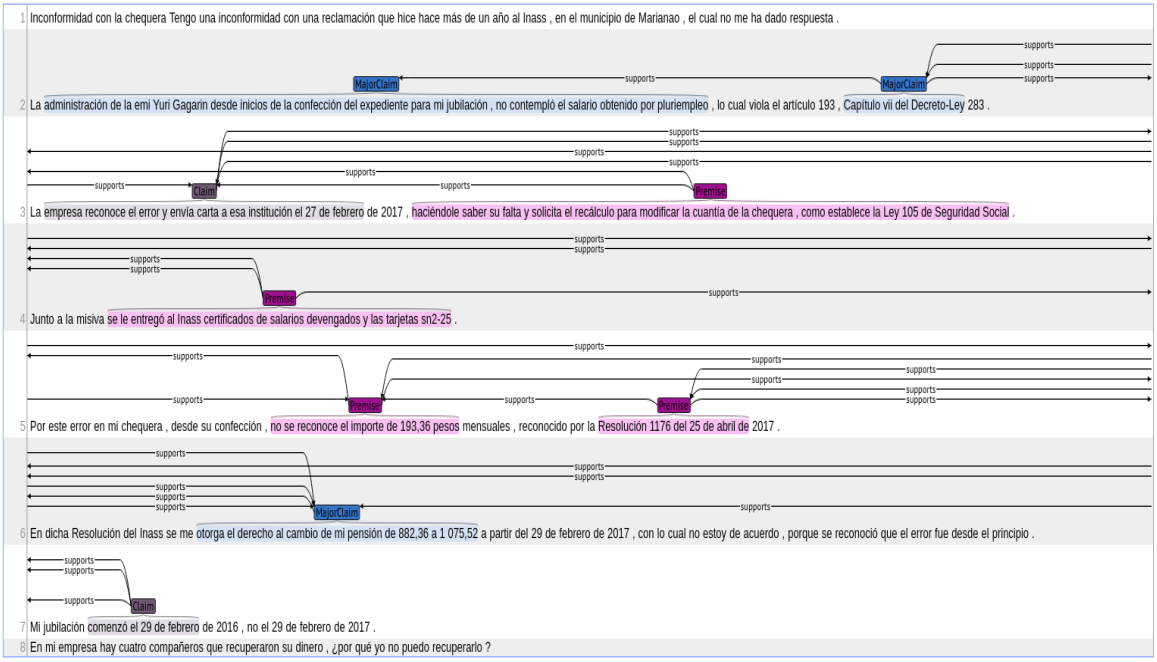

Figura 4.1: Visualización con Brat de las estructuras argumentativas.

- Inicio de UDA con dos relaciones: $atletas \quad B-Claim-attacks--1-attacks-12$

4.3. Experimentación

Para realizar la selección del modelo se utilizó el corpus de Ensayos Argumentativos. Con este se ajustaron las arquitecturas e hiperparámetros de los modelos propuestos. La mejor combinación de estos fue utilizada para el entrenamiento de los corpus restantes. Finalmente, los modelos fueron utilizados para anotar las Cartas a la Dirección.

4.3.1. Hardware

Gran parte del procesamiento se llevo a cabo en una computadora $i5$ con $8GB$ de RAM ampliada con $4GB$ de memoria *swap* [*What is memory swapping? - definition from Techopedia* 2022], aunque se requirió el uso de la plataforma Colab [*Google Colaboratory* s.f.] para el entrenamiento de algunos modelos por falta de recursos locales.

4.3.2. Segmentador de UDA

En el entrenamiento del segmentador de UDA se hicieron variaciones en la arquitectura propuesta con respecto a la presencia o no de las siguientes componentes, presentando cuatro candidatos (Tabla 4.3):

- Atributos de POS en la entrada del algoritmo (POS).
- Atributos extraídos por la CNN de la palabra (Char-CNN).
- Atributos extraídos por la LSTM bidireccional de la palabra (Char-LSTM).
- Conexiones residuales (Res).
- Capa densa final (Densa).
- Capas de normalizaciones (Norm).

Tabla 4.3: Variantes de arquitectura de los modelos de segmentación de UDA.

Modelos	POS	Char-CNN	Char-LSTM	Res	Norm	Densa
Modelo 1	×	×	×	×	×	×
Modelo 2	×	✓	✓	✓	✓	×
Modelo 3	✓	✓	✓	✓	✓	×
Modelo 4	✓	✓	✓	✓	✓	✓

En la Figura 4.2 se observan las diferentes curvas de aprendizaje de los modelos probados. Se muestra la rápida convergencia de los modelos con conexiones residuales y normalizaciones. Se muestra también la tendencia al sobreajuste en el entrenamiento entre los pasos 17-20, en donde se detiene el entrenamiento para evitar el crecimiento del error de generalización.

Las métricas 100 %F1 y 50 %F1 muestran que los modelos 1 y 2 presentan un desempeño menor que los 3 y 4. Se muestra un ligero aumento de 1 % en las 50 %F1 en el modelo 4 con respecto al modelo 3, aunque las métricas de F1 en el 3 superan a las de 4. Se considera a la segmentación como tarea principal, por lo que se selecciona como mejor modelo al 4 (Figura 4.3). Las distinciones BIOES en los nombres de tablas o métricas constituyen las métricas correspondientes a la segmentación estrictamente, mientras las que no poseen dicha distinción constituyen al proceso conjunto de segmentación y clasificación de UDAs.

El modelo seleccionado fue usado en el entrenamiento de los demás conjuntos de datos obteniendo los resultados mostrados en Tabla 4.4 y Tabla 4.5.

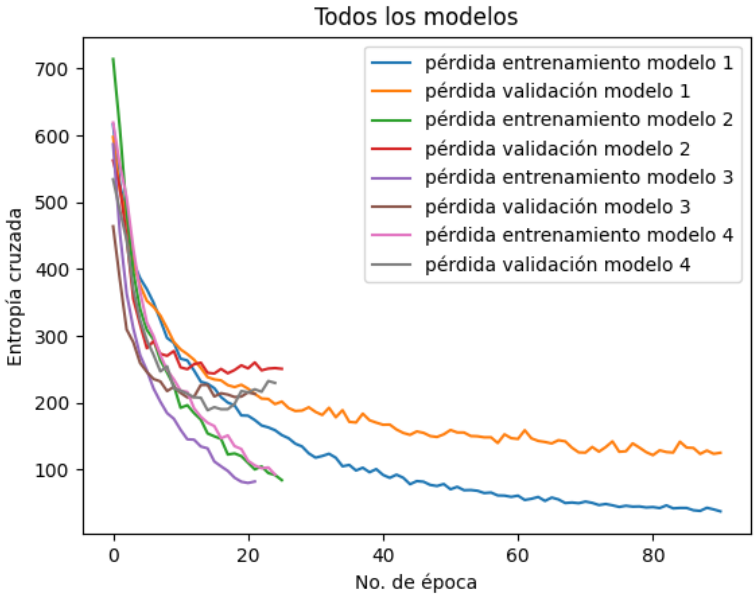

Figura 4.2: Pérdida de los modelos segmentadores.

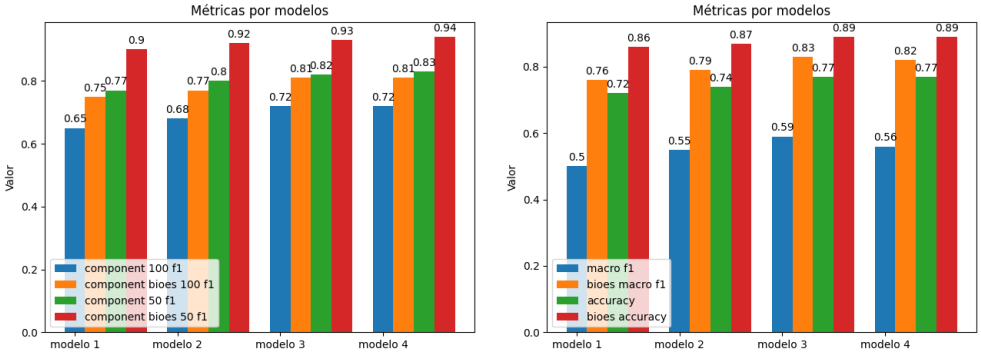

Figura 4.3: Métricas del conjunto de pruebas de los modelos segmentadores.

4.3.3. Predictor de Enlaces

Para el modelo se realizó un voto conjunto del ensamblado de 3 modelos, dado que el entrenamiento está basado en la aleatoriedad, se entrenan los modelos con los mismos datos obteniendo inferencias no necesariamente iguales. Para la selección del modelo se entrenaron diferentes variantes de arquitecturas e hiperparámetros, y al igual que en el segmentador de UDAs se realizó la selección del modelo que mejor se

Tabla 4.4: Métricas de las pruebas del segmentador de UDA.

Corpus	F1 Ponderado	Macro F1	*Accuracy*	100%F1	50%F1
Ensayos Argumentativos	0,76	0,56	0,77	0,72	0,83
CDCP	0,65	0,45	0,66	0,61	0,68
AbsTRCT	0,86	0,50	0,87	0,61	0,75

Tabla 4.5: Métricas BIOES de las pruebas del segmentador de UDA.

Corpus	F1 Ponderado	Macro F1	*Accuracy*	100%F1	50%F1
Ensayos Argumentativos	0,89	0,82	0,89	0,81	0,94
CDCP	0,95	0,56	0,96	0,82	0,93
AbsTRCT	0,90	0,79	0,91	0,66	0,82

desempeñó en el conjunto de datos de Ensayos Argumentativos. De las variaciones surgieron las siguientes propuestas (Tabla 4.6):

Tabla 4.6: Variantes de arquitectura de los modelos de predicción de enlaces.

Modelos	Atención	Pooling	*Dropout*	Tasa de aprendizaje	Paciencia	Devolver mejores
Modelo 1	×	5	0,5	0,0015	10	✓
Modelo 2	×	10	0,1	0,003	5	×
Modelo 3	✓	1	0,1	0,003	5	×
Modelo 4	✓	1	0,5	0,0015	10	✓

Las curvas de aprendizaje del proceso de entrenamiento de los modelos (Figura 4.4) muestran un nivel de sobreajuste elevado que disminuyen cuando el *dropout* aumenta. Además, se observan valores de pérdida elevados lo que significa que al modelo le cuesta ajustarse de manera satisfactoria a los datos.

Las métricas obtenidas por las diferentes versiones de los modelos (Figura 4.5) muestran que el modelo 2 constituye una opción ligeramente superior en lo correspondiente a predicción de enlace a los otros modelos. Esos hiperparámetros fueron utilizados para el entrenamiento con los demás conjuntos de datos.

En el entrenamiento del modelo en los demás conjuntos de datos se obtuvieron los resultados de las Tablas 4.7 y 4.8.

4.3.4. Acoplamiento de los modelos

Dado que la clasificación de UDAs es hecha tanto en el segmentador como en el predictor de enlaces, es necesaria la selección de cómo se va a desambiguar esta

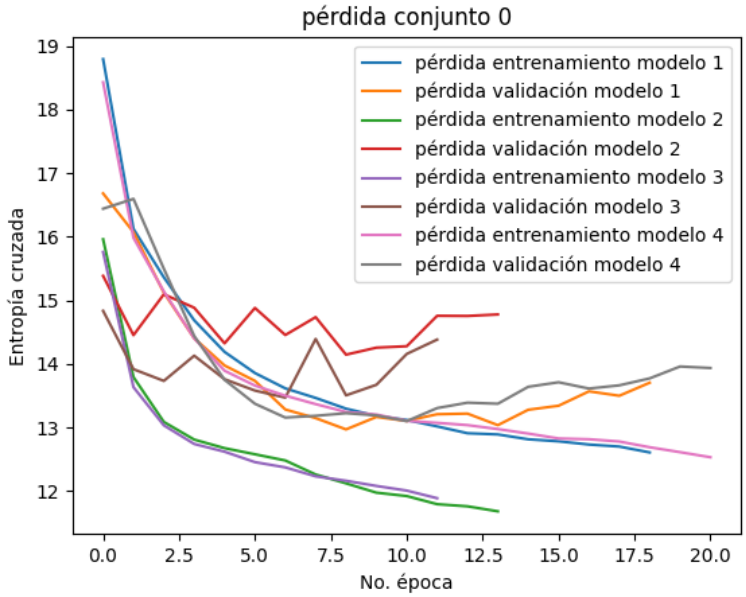

Figura 4.4: Curvas de aprendizaje de los modelos de predicción de enlaces.

Tabla 4.7: Métricas de predicción de relaciones de las pruebas del predictor de enlace.

Corpus	Macro F1	*Accuracy*	Macro F1 Enlace	*Accuracy* Enlace
Ensayos Argumentativos	0,33	0,57	0,68	0,75
CDCP	0,37	0,63	0,79	0,68
AbsTRCT	0,39	0,61	0,83	0,74

Tabla 4.8: Métricas de predicción de fuente de las pruebas del predictor de enlace.

Corpus	F1 Macro	*Accuracy*
Ensayos Argumentativos	0,48	0,60
CDCP	0,26	0,52
AbsTRCT	0,51	0,79

clasificación. En caso de seleccionar el predictor de enlaces como clasificador final, surgen varias cuestiones, como por ejemplo, que las UDAs pueden tomar varias clasificaciones o el predictor no toma el contexto del texto completo en la clasificación. Aunque la primera puede ser corregida mediante la selección de la clase más votada o

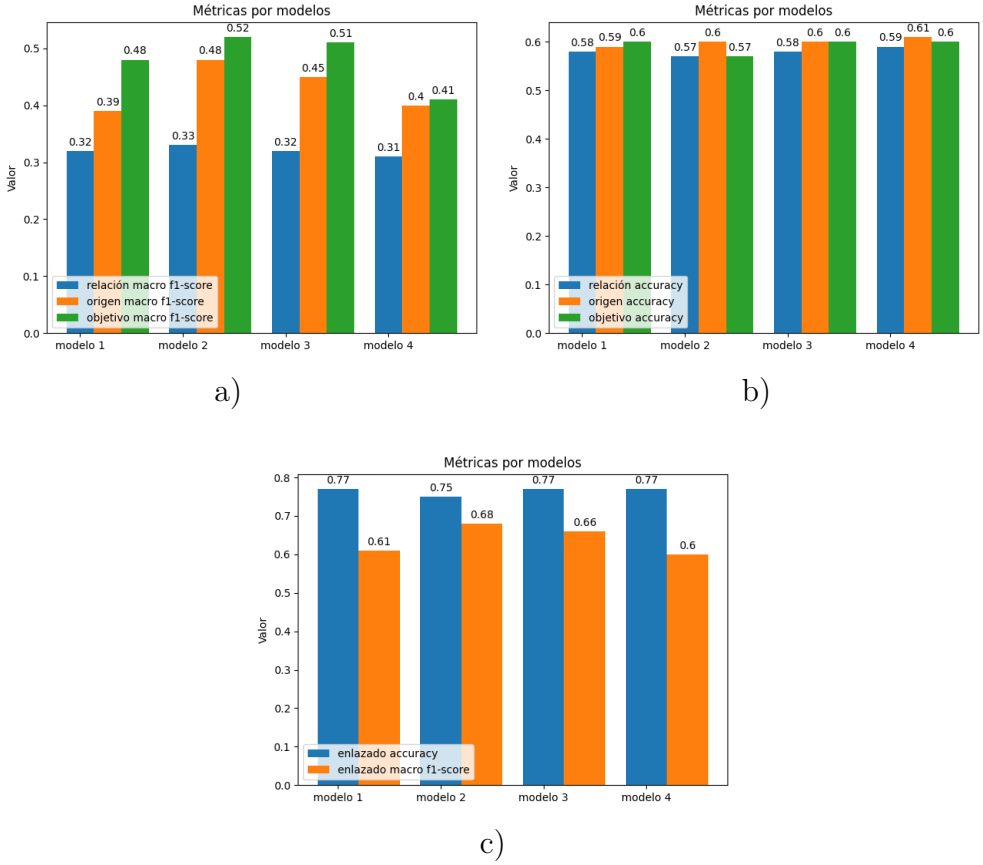

Figura 4.5: Métricas F1 de la clasificación de enlace (a), *accuracy* (b) y F1 de la predicción de enlace (c) de los modelos de predicción de enlaces.

Tabla 4.9: Métricas de predicción de objetivo de las pruebas del predictor de enlace.

Corpus	Macro F1	*Accuracy*
Ensayos Argumentativos	0,52	0,57
CDCP	0,36	0,54
AbsTRCT	0,53	0,81

algún otro criterio, la segunda presenta un mayor problema. Por esto es seleccionada la clasificación del segmentador como etiqueta final para las UDAs y el predictor es usado para la tarea de extracción y clasificación de relaciones.

4.3.5. Comparación con el estado del arte

Las comparaciones se realizan por conjuntos de datos y se muestran las métricas indicadas por los autores de cada propuesta. Cada corpus y propuesta presenta características únicas que hacen que comparaciones directas sean difíciles de realizar, por lo tanto, la información brindada muestra una comparación cualitativa y no cuantitativa en la mayoría de los casos.

Una de las principales dificultades está dada por el hecho de que las métricas calculadas son de la versión proyectada al español, lo cual contribuye a variaciones en las etiquetas finales debido al lenguaje mismo o a errores en el proceso. Otros ejemplos en la dificultad de comparar las métricas se encuentra en los enfoques tomados por las investigaciones anteriores a la hora de realizar las tareas. En algunos casos la segmentación se presenta como una tarea de clasificación BIO, o simplemente se separan por oraciones y las clasifican en argumentativas o no. En el aspecto de clasificación de las UDAs se emplean métodos como su clasificación independiente luego de ser extraída o su modelación conjunta con la segmentación. En la extracción y clasificación de relaciones se observan técnicas de optimización de problemas enteros, clasificación por SVM o también probando los posibles enlaces dos a dos independientemente.

En la comparación de métodos se seleccionaron seis métricas que evalúan las diferentes tareas de la EA. La métrica BIOES F1 se refiere a la Macro F1 de la clasificación de las etiquetas BIOES, esta constituye una medida que califica la tarea de segmentación de UDAs en el texto. La métrica Clas UDA F1 es calculada como la Macro F1 de las etiquetas BIOES junto con las etiquetas del tipo de UDA, medida que evalúa la tarea de clasificación de las UDAs. Rel Pred F1 es la medida Macro F1 de la predicción de enlaces y Rel Clas F1 la de la clasificación, estas son calculadas tomando en cuenta todos los pares seleccionados para el conjunto de datos. En la comparación ✓ significa que son directamente comparables, $*$ que el método de comparación es el mismo, pero no son usados los mismos elementos para calcular la métrica, y $\times$ que la métrica no se encontraba disponible.

Tabla 4.10: Métricas comparativas de Ensayos Persuasivos.

Corpus	Propuesto	Stab y Gurevych 2017	Niculae, Park y Cardie 2017	Galassi 2021
BIOES F1	0,82	0,85 ✓	×	×
Clas UDA F1	0,56	0,82	0,77	0,53
Rel Pred F1	0,68	0,58	0,60	0,36 *
Rel Clas F1	0,33	0,70	×	0,18 *

Tabla 4.11: Métricas comparativas de CDCP.

Corpus	Propuesto	Niculae, Park y Cardie 2017	Galassi 2021
BIOES F1	0,56	×	×
Clas UDA F1	0,45	0,73	0,79
Rel Pred F1	0,68	0,27	0,30 *
Rel Clas F1	0,37	×	0,15 *

Tabla 4.12: Métricas comparativas de AbsTRCT.

Corpus	Propuesto	Mayer, Cabrio y Villata 2020	Galassi 2021
BIOES F1	0,79	×	×
Clas UDA F1	0,50	0,88 ✓	0,91
Rel Pred F1	0,74	×	0,54 *
Rel Clas F1	0,39	0,66 *	0,70 *

4.4. Validación

Dado que las estructuras argumentativas varían en su forma en cada corpus es complejo realizar un método que evalúe de forma justa los resultados obtenidos por los diferentes modelos de manera conjunta. Una variante sería anotar las cartas con los esquemas argumentativos presentes en los conjuntos de datos, esto constituye una labor en la que se requiere personal experto y previo estudio y preparación, además de tiempo. El proceso que se lleva a cabo para realizar la validación consiste en un análisis cualitativo realizado a criterio del autor. Para esto se seleccionaron 15 pares de cartas, la carta original y la respuesta enviada a esta. Cada uno de estas 30 cartas fueron anotadas por los modelos entrenados en cada conjunto de datos y se realizó el análisis de calidad correspondiente. Los puntos por los que se llevó a cabo el análisis se resumen en los siguientes:

- ¿La UDA se extrajo correctamente?
- ¿La UDA se clasificó correctamente?
- ¿La relación se extrajo correctamente?
- ¿La relación se clasificó correctamente?

4.4.1. Análisis de Ensayos Argumentativos

Los ensayos argumentativos presentan una anotación de UDAs a un nivel de unidades de texto que pueden ser más pequeñas que oraciones y clasifican estas en las clases *MajorClaim* (MC), *Claim* (C) y *Premise* (P). Las relaciones se clasifican en de *supports* y *attacks*.

En general, se observa que se presentan problemas en la segmentación de UDAs debido al formato y dominio del texto. Las cartas presentan una estrutura en donde al final se realiza una firma poniendo información acerca del remitente, esta estructura no contribuye a la argumentación, pero el modelo en varias ocasiones detecta componentes en estas. Además de estos problemas de la estructura característica de la carta se encuentran otros. Entre los más encontrados se observa la extracción de supuestas UDAs que en su contenido no se encuentra un componente argumentativo, generalmente, estos elementos, si se expanden, pueden lograr establecer una mejor UDA.

Ejemplos malos:

- [años de explotación y]$_{MC}$: Muy corto y no informativo.
- [en cada uno de los establecimientos de nuestra Cadena de Tiendas]$_{MC}$: Incompleto, mejora incorporando elementos de la izquierda (No a todos los productos con próxima fecha de vencimiento se le aplica rebaja de precios).
- [Director División Grandes Centros]$_{MC}$ [TRD Caribe]$_{C}$: Mala clasificación y segmentación.
- [Esperamos lo antes posible una solución]$_{P}$: En contexto, no contiene información que lo haga premisa
- [no podemos permitir]$_{C}$: No establece una *claim*

Ejemplos buenos:

- [no se le puede volver a despachar, tiene que ver a la administración (si está ahí en ese momento), si no, regresar al día siguiente para que se le acredite lo sucedido]$_{MC}$
- [administración de la EMI Yuri Gagarin desde inicios de la confección del expediente para mi jubilación, no contempló el salario obtenido por pluriempleo]$_{MC}$
- pudiese [contribuir al ahorro de agua y la prestación de un mejor servicio]$_{C}$
- [es que estamos limitados de este servicio, y no desde hace un tiempo, es que nunca lo hemos tenido]$_{P}$

- [el número de carné de identidad que se encontraba en dicha base de datos correspondía a otra persona que fue reportada como fallecida]$_P$

Las relaciones anotadas por el modelo tienden a contener una gran cantidad de falsos positivos, además dado que este conjunto de datos posee un gran desbalance en las etiquetas de las relaciones favoreciendo estas a las de *supports*, el modelo no fue capaz de realizar anotaciones de *attacks*, tanto en el conjunto de original de pruebas como en las cartas.

4.4.2. Análisis de CDCP

CDCP realiza la segmentación de UDAs en la mayoría de las situaciones al separarlas por oraciones (solamente el 1% de los tokens se encuentran fuera de una UDA), estas son clasificadas en *testimony* (T), *fact* (F), *policy* (P), *reference* (R) y *value* (V). Las relaciones presentan dos tipos de relaciones *evidences* y *reasons*.

Los errores más comunes cometidos en la segmentación, dado el esquema utilizado, provienen del uso de signos de puntuación que no representan un cambio de oración, en estos casos se separa las UDA. También existen errores de clasificación incorrecta, de, por ejemplo, *testimony* que podrían ser *fact*.

Ejemplos malos:

- [. . . Bayamo, km 1 No.]$_T$ [30 A (interior), . . .]$_T$: Uso del **.** para abreviar número se toma como separador de UDA.
- [. . . ,desde el 1ro.]$_T$ [de marzo . . .]$_T$: Uso del **.** para abreviar primero se toma como separador de UDA.
- [Junto a la misiva se le entregó al Inass certificados de salarios devengados y las tarjetas sn2-25.]$_T$: Se clasifica mejor como *fact*.
- [Caridad Real Gutiérrez, Jefe de Trámites y Pensiones, Inass.]$_T$: Firma de la carta como elemento argumentaivo.

Ejemplos buenos:

- [En el 2017 se colocaron tuberías de 200, 160 y 110 mm, quedando no concluido y de continuidad para el 2018 se situó un financiamiento para construcción de registros y colocación de válvulas y ventosas.]$_F$
- [Mi jubilación comenzó el 29 de febrero de 2016, no el 29 de febrero de 2017.]$_T$
- [No se sabe cuánto queda, lo que obliga al cliente a estar haciendo cuentas constantemente.]$_F$

Las relaciones presentes disminuyen en cantidad en comparación con lo visto en los textos anotados con el modelo entrenado con Ensayos Argumentativos. Prevaleciendo las relaciones de *reasons*. A consideración del autor, la cantidad de los falsos positivos son menores que el modelo de Ensayos Argumentativos.

4.4.3. Análisis AbsTRCT

El conjunto de datos presenta un estilo de segmentación de UDAs en donde se anotan secciones de textos más grandes que en Ensayos Argumentativos, aunque no necesariamente todas las oraciones o la oración completa es considerada argumentativa. Estas se clasifican igual que Ensayos Argumentativos, aunque en este conjunto de datos se presenta un desbalance de etiquetas grande, favoreciendo a las *Premise* y las *Claim*, dejando sin representación casi a *MajorClaim* (menor del 1% de las etiquetas BIOES), lo que trajo como consecuencia que el modelo no fuera capaz de diferenciar este tipo de UDA. Las relaciones se presentaron como *partial-attack*, *attack* y *support*, influenciadas también por la poca cantidad de relaciones de *attack*.

En la clasificación de UDAs se evidencia una gran cantidad de *Premise*.

Ejemplos malos:

- [...Calle 14, apto.]$_P$ [4 entre Zapata y 23, ...]$_T$: Uso del **.** para abreviar apartamento se toma como separador de UDA.
- [, Director División Grandes Centros TRD Caribe.]$_C$: Mala clasificación con mala segmentación y detección de *claim* en firma de carta.

Ejemplos buenos:

- [Esta situación pudo haberse evitado y no podemos permitir que hechos como este ocurran pues, empañan la imagen de la institución que está destinada a brindar un servicio con calidad a la población en general y en especial a los jubilados y pensionados.]$_P$
- [Esta respuesta considera sin razón la preocupación de un lector, ¿así debe terminar la inquietud de un ciudadano, que confía en las instituciones con que cuenta la sociedad para enfrentar sus problemas?]$_C$

Las relaciones tienen la menor cantidad de elementos de los otros conjuntos de datos, proliferando las relaciones de *support*. En las clasificadas como *partial-attack* se evidenció, a consideración del autor, una baja precisión. Por ejemplo [*cierto que la responsabilidad es de todos, pero la institucional es de la Dirección de Comunales.*] ataca a [*Antonio Blanco, Director de Servicios Comunales Plaza,*].

4.4.4. Resultados

Se considera que el conjunto de datos CDCP se ajusta mejor a las características de las Cartas a la Dirección. Este presenta orígenes similares, es contenido generado por usuarios en una plataforma web, y, por lo tanto, presenta un conjunto de etiquetas de UDAs que se ajustan más a lo observado en las cartas. También las cartas presentan un alto contenido argumentativo, por lo que marcar todas las oraciones como argumentativas no constituye una fuente grande de errores, aunque esta parte puede ser mejorada. Una desventaja de este esquema sobre otros es la carencia de una clasificación de las relaciones que implique un ataque, aunque esto se cubre con el hecho de que en los conjuntos en donde existen estas los resultados son bastante pobres en ese aspecto. La cantidad y calidad de relaciones, aunque tiene bastante espacio para mejorar, es aceptable dada la dificultad del problema en EA.

Con CDCP se obtiene una visión general de la argumentación, aunque si se desea obtener algo más específico podría considerarse el modelo de Ensayos Argumentativos. La ventaja de este modelo en la extracción y clasificación de UDA es que utiliza un conjunto de etiquetas que podría considerarse universal en la argumentación y además reduce el espacio de búsqueda de oraciones a segmentos de palabras, aunque estos puedan estar sujetos a errores.

La versión de AbsTRCT constituye el modelo con menor rendimiento. La clasificación de UDAs presenta una gran desproporción hacia *Premise* dejando muchas *Claim* sin ser correctamente clasificadas. Sobre las relaciones muestra un nivel muy bajo de relaciones por documento, en relación a las que se podrían formar.

Conclusiones

En el trabajo se logró la extracción de estructuras argumentativas en los textos de las Cartas a la Dirección del periódico Granma. Para esto se extrajeron los textos de las Cartas creando un conjunto de datos conteniendo, no solo el texto de las cartas, si no, los comentarios escritos por los usuarios e información sobre la carta a la que responde, si es una respuesta. Se creó un software que puede ser utilizado no solo para el trabajo con las Cartas a la Dirección, si no que este permite un uso general para las tareas de proyección de corpus y trabajo relacionados a la EA. Este permite incorporar nuevas componentes haciendo posible una extensión simple y desacoplada. Para la anotación de las estructuras se crearon conjuntos de datos en español a partir de conjuntos anotados en inglés para poder tener datos con los cuales entrenar los modelos propuestos. Estos modelos, uno encargado de la segmentación y clasificación de las UDAs, y otro de la extracción y clasificación de las relaciones entre estas, fueron utilizados para la anotación de los textos extraídos. Los resultados obtenidos en la tarea de segmentación se encuentran al nivel del estado del arte, en las demás tareas no se encontraron comparaciones directas al enfoque tomado en la propuesta, aunque no teniendo en cuenta esto, se obtienen comparativas inferiores en la clasificación de UDAs y en la clasificación de relaciones, aunque se supera la métrica de predicción de enlace. Los resultados obtenidos del procesamiento de las Cartas a la Dirección son considerados aceptables por autor.

Recomendaciones

La principal dificultad en el trabajo fue la carencia de un conjunto de anotados sobre el tema en específico relacionado con la extracción de argumentos en la prensa. Por lo que se propone la creación de este conjunto para poder realizar una mejor validación y entrenamiento del modelo propuesto. También se considera la creación de un servicio online basado en Brat para la socialización y mejora de los resultados obtenidos. El uso de representaciones BERT ha llevado a muchas tareas de PLN a nuevos estados del arte, por lo tanto se propone investigar el uso de estos *embeddings* en el modelo. El problema principal obtenido en el modelo fue relacionado con la predicción de enlaces, un problema que tiene el modelo es la falta de contexto global del texto para hacer la predicción, por lo que se insta a la búsqueda y experimentación de métodos que tomen esto en cuenta, una variante podría ser las *Graph Neural Networks*.

Bibliografía

[1] *A fast and powerful scraping and web crawling framework*. URL: `https://scrapy.org/`.

[2] Pierre Baldi y Peter J Sadowski. «Understanding Dropout». En: *Advances in Neural Information Processing Systems*. Ed. por C.J. Burges y col. Vol. 26. Curran Associates, Inc., 2013.

[3] *Brat Rapid Annotation Tool*. URL: `https://brat.nlplab.org/`.

[4] J. Brownlee. *Better Deep Learning: Train Faster, Reduce Overfitting, and Make Better Predictions*. Machine Learning Mastery, 2018.

[5] Xiang Dai y Heike Adel. «An analysis of simple data augmentation for named entity recognition». En: *arXiv preprint arXiv:2010.11683* (2020).

[6] Jacob Devlin y col. «Bert: Pre-training of deep bidirectional transformers for language understanding». En: *arXiv preprint arXiv:1810.04805* (2018).

[7] Thomas G Dietterich. «Ensemble learning». En: *The Handbook of Brain Theory and Neural Networks* 2.1 (2002), págs. 110-125.

[8] Zi-Yi Dou y Graham Neubig. «Word Alignment by Fine-tuning Embeddings on Parallel Corpora». En: *Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. 2021.

[9] Chris Dyer, Victor Chahuneau y Noah A Smith. *A Simple, Fast, and Effective Reparameterization of IBM Model 2*. 2013, págs. 9-14. URL: `http://github.com/clab/fast`.

[10] Natalie Dykes y col. «Reconstructing arguments from noisy text». En: *Datenbank-Spektrum* 20.2 (2020), págs. 123-129.

[11] Steffen Eger, Johannes Daxenberger e Iryna Gurevych. «Neural end-to-end learning for computational argumentation mining». En: *arXiv preprint arXiv:1704.06104* (2017).

[12] Steffen Eger, Johannes Daxenberger, Christian Stab y col. «Cross-lingual argumentation mining: Machine translation (and a bit of projection) is all you need!» En: *arXiv preprint arXiv:1807.08998* (2018).

[13] Marcos Esteve, Francisco Casacuberta y Paolo Rosso. «Minería de argumentación en el Referéndum del 1 de Octubre de 2017». En: *Procesamiento del Lenguaje Natural* 65 (2020), págs. 59-66.

[14] Steven Y. Feng y col. «A Survey of Data Augmentation Approaches for NLP». En: (mayo de 2021). URL: http://arxiv.org/abs/2105.03075.

[15] Andrea Galassi. «Deep Networks and Knowledge: from Rule Learning to Neural-Symbolic Argument Mining». En: (2021).

[16] *Google Colaboratory*. URL: https://colab.research.google.com/.

[17] Theodosis Goudas y col. «Argument extraction from news, blogs, and the social web». En: *International Journal on Artificial Intelligence Tools* 24.05 (2015), pág. 1540024.

[18] Margherita Grandini, Enrico Bagli y Giorgio Visani. «Metrics for multi-class classification: an overview». En: *arXiv preprint arXiv:2008.05756* (2020).

[19] *Historical Perspectives on Argumentation - Toulmin Argument*. Oct. de 2022. URL: https://owl.purdue.edu/owl/general_writing/academic_writing/historical_perspectives_on_argumentation/toulmin_argument.html.

[20] Sepp Hochreiter y Jürgen Schmidhuber. «Long short-term memory». En: *Neural computation* 9.8 (1997), págs. 1735-1780.

[21] Diederik P Kingma y Jimmy Ba. «Adam: A method for stochastic optimization». En: *arXiv preprint arXiv:1412.6980* (2014).

[22] Eliyahu Kiperwasser y Yoav Goldberg. «Simple and accurate dependency parsing using bidirectional LSTM feature representations». En: *Transactions of the Association for Computational Linguistics* 4 (2016), págs. 313-327.

[23] John Lafferty, Andrew McCallum y Fernando CN Pereira. «Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data». En: *Proceedings of the 18th International Conference on Machine Learning 2001 (ICML 2001)*. 2001, págs. 282-289.

[24] John Lawrence y Chris Reed. «Argument Mining: A Survey». En: *Computational Linguistics* 45.4 (ene. de 2020), págs. 765-818. ISSN: 0891-2017. DOI: 10.1162/coli_a_00364. eprint: https://direct.mit.edu/coli/article-pdf/45/4/765/1847520/coli_a_00364.pdf. URL: https://doi.org/10.1162/coli%5C_a%5C_00364.

[25] Bings Liu. «Sentiment analysis and subjectivity.» En: *Handbook of natural language processing* 2.2010 (2010), págs. 627-666.

[26] Christopher D Manning. *Introduction to information retrieval*. Syngress Publishing, 2008.

[27] Tobias Mayer, Elena Cabrio y Serena Villata. «Transformer-based argument mining for healthcare applications». En: *ECAI 2020*. IOS Press, 2020, págs. 2108-2115.

[28] Tomas Mikolov y col. «Efficient estimation of word representations in vector space». En: *arXiv preprint arXiv:1301.3781* (2013).

[29] *Natural Language Toolkit*. URL: http://www.nltk.org/.

[30] Vlad Niculae, Joonsuk Park y Claire Cardie. «Argument mining with structured SVMs and RNNs». En: *arXiv preprint arXiv:1704.06869* (2017).

[31] Robert Östling y Jörg Tiedemann. «Efficient word alignment with markov chain monte carlo». En: *The Prague Bulletin of Mathematical Linguistics* (2016).

[32] Raquel Mochales Palau y Marie-Francine Moens. *Argumentation Mining: The Detection, Classification and Structuring of Arguments in Text*. 2009.

[33] *Pandas*. URL: https://pandas.pydata.org/.

[34] Jeffrey Pennington, Richard Socher y Christopher D Manning. «Glove: Global vectors for word representation». En: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, págs. 1532-1543.

[35] Chaïm Perelman y col. *The New Rhetoric: A Treatise on Argumentation*. University of Notre Dame Press, 1969. ISBN: 9780268004460. (Visitado 18-11-2022).

[36] Isaac Persing y Vincent Ng. «End-to-End Argumentation Mining in Student Essays». En: *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. San Diego, California: Association for Computational Linguistics, jun. de 2016, págs. 1384-1394. DOI: 10.18653/v1/N16-1164. URL: https://aclanthology.org/N16-1164.

[37] Slav Petrov, Dipanjan Das y Ryan McDonald. «A universal part-of-speech tagset». En: *arXiv preprint arXiv:1104.2086* (2011).

[38] *Python*. URL: https://www.python.org/.

[39] Gözde Gül Şahin y Mark Steedman. «Data augmentation via dependency tree morphing for low-resource languages». En: *arXiv preprint arXiv:1903.09460* (2019).

[40] *Scikit-Learn*. URL: https://scikit-learn.org/stable/index.html.

[41] Rico Sennrich, Barry Haddow y Alexandra Birch. «Improving neural machine translation models with monolingual data». En: *arXiv preprint arXiv:1511.06709* (2015).

[42] *Spacy · Industrial-strength natural language processing in python*. URL: https://spacy.io/.

[43] Christian Stab e Iryna Gurevych. «Parsing argumentation structures in persuasive essays». En: *Computational Linguistics* 43.3 (2017), págs. 619-659.

[44] Manfred Stede y Jodi Schneider. «Argumentation mining». En: *Synthesis Lectures on Human Language Technologies* 11.2 (2018), págs. 1-191.

[45] *Tensorflow*. URL: `http://www.tensorflow.org/`.

[46] T Tieleman y G Hilton. «RMSProp». En: *COURSERA: Neural Networks for Machine Learning* Lecture 6.5.1 (2012).

[47] Stephen E. Toulmin. *The Uses of Argument*. 2.ª ed. Cambridge University Press, 2003.

[48] Frans H Van Eemeren y Rob Grootendorst. *A systematic theory of argumentation: The pragma-dialectical approach*. Cambridge University Press, 2004.

[49] René Venegas. «Hacia una identificación automatizada de rasgos argumentativos en corpus». En: *Discurso especializado e instituciones formadoras* (2005), págs. 127-158.

[50] *Visualization with python*. URL: `https://matplotlib.org/`.

[51] Hanna M Wallach. «Conditional Random Fields: An introduction». En: *Technical Reports (CIS)* (2004), pág. 22.

[52] *What is memory swapping? - definition from Techopedia*. 2022. URL: `https://www.techopedia.com/definition/30467/memory-swapping`.

[53] David Yarowsky y Grace Ngai. «Inducing multilingual POS taggers and NP bracketers via robust projection across aligned corpora». En: *Second Meeting of the North American Chapter of the Association for Computational Linguistics*. 2001.

[54] Aston Zhang y col. «Dive into Deep Learning». En: *CoRR* abs/2106.11342 (2021). arXiv: `2106.11342`.